ONTAP Automation

ONTAP Automation

NetApp
August 29, 2024

Table of Contents

ONTAP Automation

What's new

What's new with the ONTAP REST API

NetApp periodically updates the ONTAP REST API to bring you new features, enhancements, and bug fixes.

You should also review the ONTAP Release Notes for additional information including known limitations or issues. Also see Changes to ONTAP REST API calls for any changes that might impact your automation software.

ONTAP 9.15.1

ONTAP 9.15.1 continues to expand the capabilities of the ONTAP REST API. The update for this release is relatively modest with support for two new features.

NFS over TLS

There are three new endpoints available with this feature. You can issue these API calls to retrieve all the NFS over TLS interfaces, retrieve a specific interface by UUID, and update the configuration properties for a TLS interface. Collectively these API calls provide an equivalent to the set of `vserver nfs tls interface` CLI commands.

NFS over TLS is available in ONTAP 9.15.1 as a public preview. As a preview offering, this feature is not supported for production workloads with ONTAP 9.15.1.

Windows backup applications and Unix-style symlinks

When a Windows backup application encounters a Unix-style symbolic link (symlink), the link is traversed and the data is returned by ONTAP and backed up. Beginning with ONTAP 9.15.1, you also have the option of backing up the symlink instead of the data it points to. This can provide several benefits, including improved performance of your backup applications. The endpoint `/protocols/cifs/services/{svm.uuid}` has been updated to include the new parameter `backup-symlink-enabled` in the `options` object.

ONTAP 9.14.1

The ONTAP 9.14.1 release includes over three dozen new API calls which continue to expand the capabilities of the ONTAP REST API. These endpoints support several new ONTAP features as well as updates to existing features. This release focuses primarily on security enhancements but also includes improvements to NAS, QOS, and performance metrics.

Security

There are two major security features that have been introduced with ONTAP 9.14.1. Open Authorization (OAuth 2.0) is a token-based framework that can be used to restrict access to your ONTAP storage resources. You can use it with clients that access ONTAP through the REST API. Configuration can be performed with any of the ONTAP administrative interfaces including the REST API. The ONTAP 9.14.1 release also includes support for Cisco Duo which provides two-factor authentication for SSH logins. You can configure Duo to operate at the ONTAP cluster or SVM level. In addition to these two new features, several endpoints have been added to improve control over your key stores.

FPolicy persistent storage

FPolicy provides a platform for ONTAP policy management. It provides a container for the various components or elements, such as events and the policy engine. You can now use the REST API to configure and administer a persistent store for the ONTAP FPolicy configuration and events. Each SVM can have one persistent store which is shared for the multiple policies within the SVM.

QOS options

Two endpoints have been introduced to allow you to retrieve and set QOS options for the cluster. For example, you can reserve a percentage of available system processing resources for background tasks.

Performance metrics

ONTAP maintains statistical information about the operational characteristics of the system. This information is presented in a database format consisting of tables and rows. With ONTAP 9.14.1, additional metrics data is added in several resources categories, including Fibre Channel, iSCSI, LUNs, and NVME. This additional metrics data continues to bring the ONTAP REST API closer to parity with the Data ONTAP API (ONTAPI or ZAPI).

Miscellaneous enhancements

There are several additional enhancements which can be helpful depending on your environment. These new endpoints improve access to the SAN initiators and control of the host cache settings as well as enable access to individual AutoSupport messages.

ONTAP 9.13.1

ONTAP 9.13.1 continues to expand the capabilities of the ONTAP REST API with over two dozen new API calls. These endpoints support new ONTAP features as well as enhancements to existing features. This release focuses on improvements to security, resource management, enhanced SVM configuration options, and performance metrics.

Resource tagging

You can use tags to group REST API resources. You might do this to associate related resources within a specific project or organizational group. Using tags can help to organize and track resources more effectively.

Consistency groups

ONTAP 9.13.1 continues to expand the availability of performance counter data. You can now access this type of statistical information to track historical performance and capacity for consistency groups. In addition, enhancements have been included that allow the parent-child relationships among consistency groups to be configured and managed.

DNS configuration per SVM

The existing DNS endpoints have been expanded to allow DNS domain and server configuration to be performed for individual SVMs.

EMS role configuration

The existing EMS support feature has been expanded to allow for the management of roles and the access control configuration assigned to the roles. This provides the ability to limit or filter the events and messages based on the role configuration.

Security

You can use the REST API to configure the time-based one-time password (TOTP) profiles for accounts that sign in and access ONTAP using SSH. In addition, the key manager endpoints have been expanded to provide a restore operation from a specified key management server.

CIFS configuration per SVM

The existing CIFS endpoints have been expanded to allow the configuration for a specific SVM to be updated.

S3 bucket rules

The existing S3 bucket endpoints have been expanded to include a rule definition. Each rule is a list objects and defines the set of actions to be performed on an object within the bucket. Collectively these rules allow you to better manage the lifecycle of your S3 buckets.

ONTAP 9.12.1

ONTAP 9.12.1 continues to expand the capabilities of the ONTAP REST API with over forty new API calls. These endpoints support new ONTAP features as well as enhancements to existing features. This release focuses on improvements to security and NAS capabilities.

Security enhancements

Amazon Web Services includes a key management service that provides secure storage for keys and other secrets. You can access this service through the REST API to allow ONTAP to securely store its encryption keys in the cloud. In addition, you can create and list the authentication keys used with NetApp Storage Encryption.

Active Directory

You can manage the Active Directory accounts defined for an ONTAP cluster. This includes creating new accounts as well as displaying, updating, and deleting accounts.

CIFS group policies

The REST API has been enhanced to support the creation and management of CIFS group policies. The configuration information is available and administered through group policy objects that are applied to all or specific SVMs.

ONTAP 9.11.1

ONTAP 9.11.1 continues to expand the capabilities of the ONTAP REST API with nearly a hundred new API calls. These endpoints support the new ONTAP features as well as enhancements to existing features. This release focuses on supporting customer migration to the ONTAP REST API from the Data ONTAP API (ONTAPI or ZAPI).

Granular RBAC

The ONTAP role-based access control (RBAC) capability has been enhanced to provide additional granularity. You can use the traditional roles or create new custom roles as needed through the REST API. Every role is associated with one or more privileges, each of which identifies a REST API call or CLI command along with the access level. New access levels are available for REST roles, such as `read_create` and `read_modify`. This enhancement provides parity with the Data ONTAP API (ONTAPI or ZAPI) and supports customer migration to the REST API. See RBAC security for more information.

Performance counters

Previous ONTAP releases have maintained statistical information about the operational characteristics of the system. With the 9.11.1 release, this information has been enhanced and is now available through the REST API. An administrator or automated process can access the data to determine system performance. The statistical information, as maintained by the counter manager subsystem, is presented in a database format using tables and rows. This enhancement brings the ONTAP REST API closer to parity with the Data ONTAP API (ONTAPI or ZAPI).

Aggregate management

The management of ONTAP storage aggregates has been enhanced. You can use the updated REST endpoints to move aggregates online and offline as well as manage the spares.

IP subnet capability

The ONTAP networking capability has been expanded to include support for IP subnets. The REST API provides access to the configuration and management of the IP subnets within an ONTAP cluster.

Multiple administrator verification

The multiple administrator verification feature provides a flexible authorization framework for protecting access to ONTAP commands or operations. You can define rules that identify the restricted commands. When a user requests access to a specific command, approval can be granted by multiple ONTAP administrators as appropriate.

SnapMirror enhancements

The SnapMirror capability has been enhanced in several areas including scheduling. The SnapVault relationship parity has been added in a DP relationship with ONTAP 9.11.1 Also, the throttle feature available with the REST API has reached parity with the Data ONTAP API (ONTAPI or ZAPI). Related to this, support is available to create and manage bulk snapshot copies.

Storage pools

Several endpoints have been added to provide access to the ONTAP storage pools. Support is included for creating and listing the storage pools in a cluster as well as updating and deleting specific pools by ID.

Name services cache support

ONTAP name services has been enhanced to support caching which improves performance and resiliency. Configuration of the name services cache can now be accessed through the REST API. Settings can be applied at multiple levels including: hosts, unix-users, unix-groups, and netgroups.

ONTAPI reporting tool

The ONTAPI reporting tool helps customers and partners identify the ONTAPI usage in their environment. In addition to the Python software, there is also a video and evolving support in the NetApp Lab on Demand. This tool provides another resource when migrating from ONTAPI to the ONTAP REST API.

ONTAP 9.10.1

ONTAP 9.10.1 continues to expand the capabilities of the ONTAP REST API. Over a hundred new endpoints have been added to support new ONTAP features as well as enhancements to existing features. A summary of the REST API enhancements is presented below.

Application consistency group

A consistency group is a set of volumes that are grouped together when performing certain operations such as a snapshot. This feature extends the same crash consistency and data integrity implicit with single-volume operations across a set of volumes. It is valuable for large multi-volume workload applications.

SVM migration

You can migrate an SVM from a source cluster to a destination cluster. The new endpoints provide complete control, including the ability to pause, resume, retrieve status, and abort a migration operation.

File cloning and management

Volume-level file cloning and management have been enhanced. New REST endpoints support file move, copy, and split operations.

Improved S3 auditing

Auditing of the S3 events is a security improvement allowing you to track and log certain S3 events. An S3 audit event selector can be set on a per SVM per bucket basis.

Ransomware defense

ONTAP detects files potentially containing a ransomware threat. You can retrieve a list of these suspect files as well as remove them from a volume.

Miscellaneous security enhancements

There are several general security enhancements that expand existing protocols and introduce new capabilities. Improvements have been made to IPSEC, key management, SSH configuration, and file permissions.

CIFS domains and local groups

Support for CIFS domains has been added at the cluster and SVM level. You can retrieve the domain configuration as well as create and remove preferred domain controllers.

Expanded volume analytics

Volume analytics and metrics have been expanded through additional endpoints to support top files, directories, and users.

Support enhancements

Support has been enhanced through several new features. Automatic update can keep your ONTAP systems current by downloading and applying the latest software updates. You can also retrieve and manage the memory core dumps generated by a node.

ONTAP 9.9.1

ONTAP 9.9.1 continues to expand the capabilities of the ONTAP REST API. There are new API endpoints for existing ONTAP features, including SAN port sets and vServer file directory security. In addition, endpoints have been added to support new ONTAP 9.9.1 features and enhancements. And the related documentation has also been improved. A summary of the enhancements is presented below.

Mapping ONTAPI to the ONTAP 9 REST API

To help you transition your ONTAP automation code to the REST API, NetApp provides API mapping documentation. This reference includes a list of ONTAPI calls and the REST API equivalent for each. The mapping document has been updated to include the new ONTAP 9.9.1 API end points. See ONTAPI to REST API mapping for more information.

API endpoints for new ONTAP 9.9.1 core features

Support for new ONTAP 9.9.1 features that are not available through the ONTAPI API has been added to the REST API. This includes support for nested igroups and Google Cloud Key Management Services.

Improved support for transitioning to REST from ONTAPI

More of the legacy ONTAPI calls now have corresponding REST API equivalents. This includes local Unix users and groups, management of NTFS file security without the need for a client, SAN port sets, and volume space attributes. These changes are also included in the updated ONTAPI to REST mapping documentation.

Enhanced online documentation

The ONTAP online documentation reference page now includes labels indicating the ONTAP release when each REST endpoint or parameter was introduced, including those new with ONTAP 9.9.1.

ONTAP 9.8

ONTAP 9.8 greatly expands the breadth and depth of the ONTAP REST API. It includes several new features which enhance your ability to automate the deployment and management of ONTAP storage systems. In addition, support has been improved for assisting with the transition to REST from the legacy ONTAPI API.

Mapping ONTAPI to the ONTAP 9 REST API

To help you update your ONTAPI automation, NetApp provides a list of ONTAPI calls that require one or more input parameters, along with a mapping of those calls to the equivalent ONTAP 9 REST API call. See ONTAPI to REST API mapping for more information.

API endpoints for new ONTAP 9.8 core features

Support for the new core ONTAP 9.8 features not available through ONTAPI has been added to the REST API. This includes REST API support for ONTAP S3 buckets and services, SnapMirror Business Continuity, and File System Analytics.

Expanded support for enhanced security

Security has been enhanced through the support of several services and protocols, including Azure Key Vault, Google Cloud Key Management Services, IPSec, and Certificate Signing Requests.

Enhancements to improve simplicity

ONTAP 9.8 delivers more efficient and modern workflows using the REST API. For example, oneclick firmware updates are now available for several different types of firmware.

Enhanced online documentation

The ONTAP online documentation page now includes labels indicating the ONTAP release that each REST endpoint or parameter was introduced, including those new in 9.8.

Improved support for transitioning to REST from ONTAPI

More legacy ONTAPI calls now have corresponding REST API equivalents. Documentation is also available to help identify which REST endpoint should be used in place of an existing ONTAPI call.

Expanded performance metrics

Performance metrics for the REST API have been expanded to include several new storage and network objects.

ONTAP 9.7

ONTAP 9.7 extends the functional scope of the ONTAP REST API by introducing three new resource categories, each with several REST endpoints:

- NDMP
- Object store
- SnapLock

ONTAP 9.7 also introduces one or more new REST endpoints in several of the existing resource categories:

- Cluster
- NAS
- Networking
- NVMe

- SAN
- Security
- Storage
- Support

ONTAP 9.6

ONTAP 9.6 greatly extends the REST API support originally introduced in ONTAP 9.4. The ONTAP 9.6 REST API supports most ONTAP configuration and administration tasks.

REST APIs in ONTAP 9.6 include the following key areas and many more:

- Cluster setup
- Protocol configuration
- Provisioning
- Performance monitoring
- Data protection
- Application aware data management

Changes to ONTAP REST API calls

NetApp continues to enhance and update the ONTAP REST API with each major product release. These updates can occasionally include changes to existing API calls, such as the parameters and default values used. These changes can affect software that accesses the REST API.

Changes to existing ONTAP REST API calls

Any changes to the existing API calls can affect software that accesses the REST API. You should review the list of changes in the table below to determine if there is an impact to your ONTAP automation environment. Each entry includes the applicable API endpoint, a description of the change, and the ONTAP release it was introduced.

Endpoint	Description of change	ONTAP release
`/security/authentication/duo/groups` `/security/authentication/duo/profiles`	The field **_links** in the response was removed from the duogroup for these endpoints. There is no recommended customer action or workaround. The field is expected to be added back in a future ONTAP release.	9.15.1

ONTAP REST API reference documentation errors

As NetApp enhances and updates the ONTAP REST API, sometimes errors can be introduced in the online reference documentation. These errors can create confusion when using the API but generally don’t impact or disrupt your ONTAP automation software or environment.

You should review the list of errors in the table below. This will help you better understand and navigate the ONTAP REST API reference documentation. Each entry includes the applicable API endpoint, a description of the error, and the ONTAP release it was introduced.

Endpoint	Description of change	ONTAP release
`/storage/quota/reports`	The REST API documentation for the endpoint indicates that **specifier** is a valid field. However, the quota specifier is not supported with this endpoint. There is no recommended customer action or workaround. This field will be removed from the API documentation in a future ONTAP release.	9.6

Related information

What's new with the ONTAP REST API

Get started

Understand the ONTAP automation options

There are several options available to automate the deployment and administration of your ONTAP storage systems.

ONTAP REST API

Beginning with ONTAP 9.6, ONTAP includes an expansive REST API that provides the foundation for automating the deployment and administration of your storage systems. Since then the REST API has continued to expand and mature. It now provides the preferred and strategic option when automating the administration of your ONTAP deployments.

Accessing the REST API natively

You can access the ONTAP REST API directly using any programming language that supports a REST client. Popular language choices include Python, PowerShell, and Java.

Migrating legacy ONTAPI code to use REST

The ONTAPI API (Zephyr API or ZAPI) is the original set of proprietary calls included with the NetApp ONTAP software to support the automation of your data storage administration and management tasks. The API is part of the NetApp Manageability SDK. It is expected the ONTAPI interface will be disabled in future versions of ONTAP. If you have existing code using the ONTAPI API, you should plan to migrate away from ONTAPI. NetApp provides support for converting your code to use the newer ONTAP REST API. See Migrate from ONTAPI to the REST API for more information.

Client software toolkits

NetApp provides client toolkits that abstract the ONTAP REST API and make it easier to create automation code. You should choose one appropriate for your development language and environment.

Python client library

The Python client library is a package you can use when writing scripts to access the ONTAP REST API. It provides support for several underlying services, including connection management, asynchronous request processing, and exception handling. By using the Python client library, you can quickly develop robust code to support your ONTAP automation goals. See Python client library for more information.

PowerShell toolkit

You can use the NetApp.ONTAP PowerShell Toolkit to automate the administration of an ONTAP cluster from a Windows host. See Overview of the PowerShell Toolkit for more information.

Automation frameworks

You can create and deploy automation code using one of several frameworks

Ansible

Ansible is an open-source software tool that supports provisioning, configuration management, and application deployment. Since its release and subsequent acquisition by RedHat, it has continued to grow in popularity. NetApp provides Ansible-certified modules that customers can use to automate the administration of their ONTAP storage systems. See Learn more and NetApp Ansible DevOps Solutions for additional information.

BlueXP automation catalog

The NetApp BlueXP automation catalog is available through the BlueXP web user interface. The catalog provides access to packaged solutions that can help you to automate the deployment and integration of ONTAP with other products. See NetApp automation for documentation and more information.

How to access the ONTAP REST API

You can access the ONTAP REST API in several different ways.

Network considerations

You can connect to the REST API through the following interfaces:

- Cluster management LIF
- Node management LIF
- SVM management LIF

The LIF you choose to use must be configured to support the HTTPS management protocol. Also, the firewall configuration in your network must allow the HTTPS traffic.

You should always use a cluster management LIF. This will load balance the API requests across all the nodes and avoid nodes that are offline or experiencing connectivity issues. If you have multiple cluster management LIFs configured, they are all equivalent regarding access to the REST API.

ONTAP API online documentation page

The ONTAP API online documentation page provides an access point when using a web browser. In addition to providing a way to execute individual API calls directly, the page includes a detailed description of the API, including input parameters and other options for each call. The API calls are organized into functional categories. See Summary of the REST resources for more information.

The format of the URL used to access the documentation page for the most recent version of the API is:

```
https://<cluster_mgmt_ip_address>/docs/api
```

Custom software and tools

You can access the ONTAP API using any of several different programming languages and tools. Popular choices include Python, Java, Curl, and PowerShell. A program, script, or tool that uses the API acts as a REST web services client. Using a programming language enables a deeper understanding of the API and provides an opportunity to automate the ONTAP administration.

The format of the base URL used to directly access the most recent version of the API is:

```
https://<cluster_mgmt_ip_address>/api
```

To access a specific API version where multiple versions are supported, the format of the URL is:

```
https://<cluster_mgmt_ip_address>/api/v1
```

Your first ONTAP REST API call

You can issue a simple curl command to get started using the ONTAP REST API and confirm its availability.

Before you begin

In addition to having the curl utility available on your workstation, you need the following:

- IP address or FQDN of the ONTAP cluster management LIF
- ONTAP credentials for an account with authority to access the ONTAP REST API

> If your credentials include special characters, you need to format them in a way that is acceptable to curl based on the shell you are using. For example, you can insert a backslash before each special character or wrap the entire credentials string in double quotes.

Steps

1. At the command line interface of your local workstation, issue the following command:

```
curl --request GET \
"https://$FQDN_IP/api/cluster?fields=version" \
--user username:password
```

Example

```
curl --request GET "https://10.29.186.132/api/cluster?fields=version" --user
admin:david123
```

After you finish

The ONTAP version information is displayed in a JSON format.

ONTAP REST API lab resources

NetApp provides a lab environment for you to test the ONTAP REST API and other related automation technologies.

The Lab on Demand is available to NetApp customers and partners. You'll need valid credentials to sign in and begin using the lab resources. You can search the lab for *REST* or other technologies as needed.

Also review Preparing the Lab on Demand to run the sample scripts to get started.

ONTAP REST API

REST implementation details

REST web services foundation

Representational State Transfer (REST) is a style for creating distributed web applications. When applied to the design of a web services API, it establishes a set of technologies for exposing server-based resources and managing their states. It uses mainstream protocols and standards to provide a flexible foundation for administering ONTAP clusters.

While REST establishes a common set of technologies and best practices, the details of each API can vary based on the choices made during development. You should be aware of the design characteristics of the ONTAP REST API before using it with a live deployment.

Resources and state representation

Resources are the basic components of a web-based system. When creating a REST web services application, early design tasks include:

- Identification of system or server-based resources

  Every system uses and maintains resources. A resource can be a file, business transaction, process, or administrative entity. One of the first tasks in designing an application based on REST web services is to identify the resources.

- Definition of resource states and associated state operations

  Resources are always in one of a finite number of states. The states, as well as the associated operations used to affect the state changes, must be clearly defined.

URI endpoints

Every REST resource must be defined and made available using a well-defined addressing scheme. The endpoints where the resources are located and identified use a Uniform Resource Identifier (URI). The URI provides a general framework for creating a unique name for each resource in the network. The Uniform Resource Locator (URL) is a type of URI used with web services to identify and access resources. Resources are typically exposed in a hierarchical structure similar to a file directory.

HTTP messages

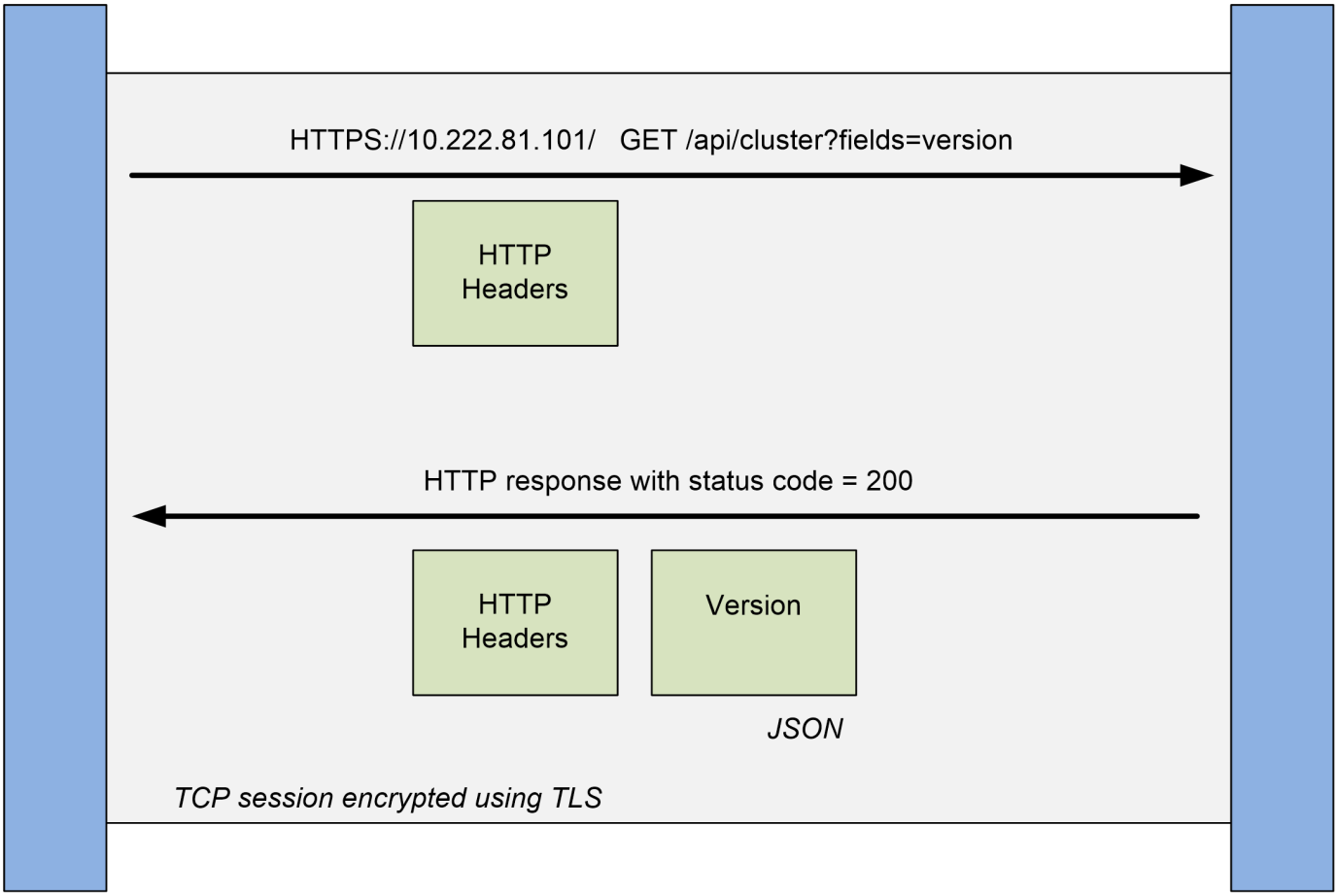
Hypertext Transfer Protocol (HTTP) is the protocol used by the web services client and server to exchange request and response messages about the resources. As part of designing a web services application, HTTP methods are mapped to the resources and corresponding state management actions. HTTP is stateless. Therefore, to associate a set of related requests and responses as part of one transaction, additional information must be included in the HTTP headers carried with the request and response data flows.

JSON formatting

Although information can be structured and transferred between a web services client and server in several ways, the most popular option is JavaScript Object Notation (JSON). JSON is an industry standard for representing simple data structures in plain text and is used to transfer state information describing the resources. The ONTAP REST API uses JSON to format the data carried in the body of each HTTP request and response.

Typical REST API transaction

Every API transaction consists of an HTTP request and the associated response. This illustration shows how to retrieve the version of the ONTAP software used by the cluster.



HTTP request

The request sent from the client to the server consists of the following:

- GET verb
- URL path for the cluster
- Query parameter (fields)
- Request headers, including authorization

HTTP response

The response sent from the server to the client consists of the following:

- Status code 200
- Response headers
- Response body containing the cluster software version

Basic operational characteristics

While REST establishes a common set of technologies and best practices, the details of each API can vary based on the design choices.

Request and response API transaction

Every REST API call is performed as an HTTP request to the ONTAP system which generates an associated response to the client. This request/response pair is considered an API transaction. Before using the API, you should be familiar with the input variables available to control a request and the contents of the response output.

Support for CRUD operations

Each of the resources available through the ONTAP REST API is accessed based on the CRUD model:

- Create
- Read
- Update
- Delete

For some of the resources, only a subset of the operations is supported. You should review the ONTAP API documentation page at your ONTAP cluster for more information about each resource.

Object identifiers

Each resource instance or object is assigned a unique identifier when it is created. In most cases, the identifier is a 128-bit UUID. These identifiers are globally unique within a specific ONTAP cluster. After issuing an API call that creates a new object instance, a URL with the associated id value is returned to the caller in the location header of the HTTP response. You can extract the identifier and use it on subsequent calls when referring to the resource instance.

The content and internal structure of the object identifiers can change at any time. You should only use the identifiers on the applicable API calls as needed when referring to the associated objects.

Object instances and collections

Depending on the resource path and HTTP method, an API call can apply to a specific object instance or a collection of objects.

Synchronous and asynchronous operations

There are two ways that ONTAP performs an HTTP request received from a client.

Synchronous processing

ONTAP performs the request immediately and responds with an HTTP status code of 200 or 201 if it is successful.

Every request using the methods GET, HEAD, and OPTIONS is always performed synchronously. In addition, requests that use POST, PATCH, and DELETE are designed to run synchronously if they are expected to complete in less than two seconds.

Asynchronous processing

If an asynchronous request is valid, ONTAP creates a background task to process the request and a job object to anchor the task. The 202 HTTP status is returned to the caller along with the job object. To determine final success or failure, you must retrieve the state of the job.

Requests that use the methods POST, PATCH, and DELETE are designed to run asynchronously if they are expected to take more than two seconds to complete.

The `return_timeout` query parameter is available with asynchronous API calls and can convert an asynchronous call to complete synchronously. Refer to Asynchronous processing using the Job object for more information.

Security

The security provided with the REST API is based primarily on the existing security features available with ONTAP. The following security is used by the API:

Transport Layer Security

All traffic sent over the network between the client and ONTAP LIF is typically encrypted using TLS, based on the ONTAP configuration settings.

Client authentication

The same authentication options available with ONTAP System Manager and the Network Manageability SDK can also be used with the ONTAP REST API.

HTTP authentication

At an HTTP level, for example when accessing the ONTAP REST API directly, there are two authentication options as described below. In each case, you need to create an HTTP authorization header and include it with each request.

Option	Description
HTTP basic authentication	The ONTAP username and password are concatenated with a colon. The string is converted to base64 and included in the request header.
OAuth 2.0	Beginning with ONTAP 9.14, you can request an access token from an external authorization server and include it as a bearer token in the request header.

For more details about OAuth 2.0 and how it is implemented in ONTAP, see Overview of the ONTAP OAuth 2.0 implementation. Also see Prepare to use the workflows below at this site.

ONTAP authorization

ONTAP implements a role-based authorization model. The account you use when accessing the ONTAP REST API or API documentation page should have the proper authority.

Input variables controlling an API request

You can control how an API call is processed through parameters and variables set in the HTTP request.

HTTP methods

The HTTP methods supported by the ONTAP REST API are shown in the following table.

Not all the HTTP methods are available at each of the REST endpoints. Also, both PATCH and DELETE can be used on a collection. See *Object references and access* for more information.

HTTP method	Description
GET	Retrieves object properties on a resource instance or collection.
POST	Creates a new resource instance based on the supplied input.
PATCH	Updates an existing resource instance based on the supplied input.
DELETE	Deletes an existing resource instance.
HEAD	Effectively issues a GET request but only returns the HTTP headers.
OPTIONS	Determine what HTTP methods are supported at a specific endpoint.

Path variables

The endpoint path used with each REST API call can include various identifiers. Each ID corresponds to a specific resource instance. Examples include cluster ID and SVM ID.

Request headers

You must include several headers in the HTTP request.

Content-type

If the request body includes JSON, this header must be set to `application/json`.

Accept

This header should be set to `application/hal+json`. If it is instead set to `application/json` none of the HAL links will be returned except a link needed to retrieve the next batch of records. If the header is anything else aside from these two values, the default value of the `content-type` header in the response will be `application/hal+json`.

Authorization

Basic authentication must be set with the user name and password encoded as a base64 string. For example:

```
Authorization: Basic YWRtaW46cGV0ZXJzb24=.
```

Request body

The content of the request body varies depending on the specific call. The HTTP request body consists of one of the following:

- JSON object with input variables
- Empty JSON object

Filtering objects

When issuing an API call with the GET method, you can limit or filter the returned objects based on any attribute using a query parameter.

Parsing and interpreting query parameters

A set of one or more parameters can be appended to the URL string beginning after the `?` character. If more than one parameter is provided, the query parameters are split based on the `&` character. Each key and value in the parameter are split at the = character.

For example, you can specify an exact value to match using the equal sign:

```
<field>=<value>
```

For a more complex query, the additional operator is placed after the equal sign. For example, to select the set of objects based on a specific field that is greater than or equal to some value, the query would be:

```
<field>=>=<value>
```

Filtering operators

In addition to the examples provided above, additional operators are available to return objects over a range of values. A summary of the filtering operators supported by the ONTAP REST API is shown in the table below.

Any fields that are not set are generally excluded from matching queries.

Operator	Description
=	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
!	Not equal to
*	Greedy wildcard

You can also return a collection of objects based on whether a specific field is set or not set by using the `null` keyword or its negation `!null` as part of the query.

Workflow examples

Some examples are included below from the REST API workflows at this site.

- List disks

  Filter based on the `state` variable to select the spare disks.

Requesting specific object fields

By default, issuing an API call using GET returns only the attributes that uniquely identify the object or objects, along with a HAL self link. This minimum set of fields acts as a key for each object and varies based on the object type. You can select additional object properties using the `fields` query parameter in the following ways:

- Common or standard fields

  Specify `fields=*` to retrieve the most commonly used object fields. These fields are typically maintained in local server memory or require little processing to access. These are the same properties returned for an object after using GET with a URL path key (UUID).

- All fields

  Specify `fields=**` to retrieve all the object fields, including those requiring additional server processing to access.

- Custom field selection

  Use `fields=<field_name>` to specify the exact field you want. When requesting multiple fields, the values must be separated using commas without spaces.

As a best practice, you should always identify the specific fields you want. You should only retrieve the set of common fields or all fields when needed. Which fields are classified as common, and returned using `fields=*`, is determined by NetApp based on internal performance analysis. The classification of a field might change in future releases.

Sorting objects in the output set

The records in a resource collection are returned in the default order defined by the object. You can change the order using the `order_by` query parameter with the field name and sort direction as follows:

```
order_by=<field name> asc|desc
```

For example, you can sort the type field in descending order followed by id in ascending order:

```
order_by=type desc, id asc
```

Note the following:

- If you specify a sort field but don’t provide a direction, the values are sorted in ascending order.
- When including multiple parameters, you must separate the fields with a comma.

Pagination when retrieving objects in a collection

When issuing an API call using GET to access a collection of objects of the same type, ONTAP attempts to return as many objects as possible based on two constraints. You can control each of these constraints using additional query parameters on the request. The first constraint reached for a specific GET request terminates the request and therefore limits the number of records returned.

If a request ends before iterating over all the objects, the response contains the link needed to retrieve the next batch of records.

Limiting the number of objects

By default, ONTAP returns a maximum of 10,000 objects for a GET request. You can change this limit using the `max_records` query parameter. For example:

```
max_records=20
```

The number of objects actually returned can be less than the maximum in effect, based on the related time constraint as well as the total number of objects in the system.

Limiting the time used to retrieve the objects

By default, ONTAP returns as many objects as possible within the time allowed for the GET request. The default timeout is 15 seconds. You can change this limit using the `return_timeout` query parameter. For example:

```
return_timeout=5
```

The number of objects actually returned can be less than the maximum in effect, based on the related constraint on the number of objects as well as the total number of objects in the system.

Narrowing the result set

If needed, you can combine these two parameters with additional query parameters to narrow the result set. For example, the following returns up to 10 ems events generated after the specified time:

```
time=> 2018-04-04T15:41:29.140265Z&max_records=10
```

You can issue multiple requests to page through the objects. Each subsequent API call should use a new time value based on the latest event in the last result set.

Size properties

The input values used with some API calls as well as certain query parameters are numeric. Rather than provide an integer in bytes, you can optionally use a suffix as shown in the following table.

Suffix	Description
KB	KB Kilobytes (1024 bytes) or kibibytes
MB	MB Megabytes (KB x 1024 bytes) or mebibytes
GB	GB Gigabytes (MB x 1024 bytes) or gibibytes
TB	TB Terabytes (GB x 1024 bytes) or tebibytes
PB	PB Petabytes (TB x 1024 bytes) or pebibytes

Related information

- Object references and access

Interpreting an API response

Each API request generates a response back to the client. You should examine the response to determine whether it was successful and retrieve additional data as needed.

HTTP status code

The HTTP status codes used by the ONTAP REST API are described below.

Code	Reason phrase	Description
200	OK	Indicates success for calls that do not create a new object.
201	Created	An object is successfully created. The location header in the response includes the unique identifier for the object.
202	Accepted	A background job has been started to perform the request, but has not completed yet.
400	Bad request	The request input is not recognized or is inappropriate.
401	Unauthorized	User authentication has failed.
403	Forbidden	Access is denied due to an authorization error.
404	Not found	The resource referred to in the request does not exist.
405	Method not allowed	The HTTP method in the request is not supported for the resource.
409	Conflict	An attempt to create an object failed because a different object must be created first or the requested object already exists.
500	Internal error	A general internal error occurred at the server.

Response headers

Several headers are included in the HTTP response generated by the ONTAP.

Location

When an object is created, the location header includes the complete URL to the new object including the unique identifier assigned to the object.

Content-type

This will normally be `application/hal+json`.

Response body

The content of the response body resulting from an API request differs based on the object, processing type, and the success or failure of the request. The response is always rendered in JSON.

- Single object

  A single object can be returned with a set of fields based on the request. For example, you can use GET to retrieve selected properties of a cluster using the unique identifier.

- Multiple objects

  Multiple objects from a resource collection can be returned. In all cases, there is a consistent format used, with `num_records` indicating the number of records and records containing an array of the object instances. For example, you can retrieve the nodes defined in a specific cluster.

- Job object

If an API call is processed asynchronously, a Job object is returned which anchors the background task. For example, the PATCH request used to update the cluster configuration is processed asynchronously and returns a Job object.

- Error object

  If an error occurs, an Error object is always returned. For example, you will receive an error when attempting to change a field not defined for a cluster.

- Empty JSON object

  In certain cases, no data is returned and the response body includes an empty JSON object.

HAL linking

The ONTAP REST API uses HAL as the mechanism to support Hypermedia as the Engine of Application State (HATEOAS). When an object or attribute is returned that identifies a specific resource, a HAL-encoded link is also included allowing you to easily locate and determine additional details about the resource.

Errors

If an error occurs, an error object is returned in the response body.

Format

An error object has the following format:

```
"error": {
"message": "<string>",
"code": <integer>[,
"target": "<string>"]
}
```

You can use the code value to determine the general error type or category, and the message to determine the specific error. When available, the target field includes the specific user input associated with the error.

Common error codes

The common error codes are described in the following table. Specific API calls can include additional error codes.

Code		Description
1	409	An object with the same identifier already exists.
2	400	The value for a field has an invalid value or is missing, or an extra field was provided.
3	400	The operation is not supported.
4	405	An object with the specified identifier cannot be found.

Code		Description
6	403	Permission to perform the request is denied.
8	409	The resource is in use.

Asynchronous processing using the Job object

After issuing an API request that is designed to run asynchronously, a job object is always created and returned to the caller. The job describes and anchors a background task that processes the request. Depending on the HTTP status code, you must retrieve the state of the job to determine if the request was successful.

Refer to API reference to determine which API calls are designed to be performed asynchronously.

Controlling how a request is processed

You can use the `return_timeout` query parameter to control how an asynchronous API call is processed. There are two possible outcomes when using this parameter.

Timer expires before the request completes

For valid requests, ONTAP returns a 202 HTTP status code along with the job object. You must retrieve the state of the job to determine if the request completed successfully.

Request is completed before the timer expires

If the request is valid and completes successfully before the time expires, ONTAP returns a 200 HTTP status code along with the job object. Because the request is completed synchronously, as indicated by the 200, you do not need to retrieve the job state.

> The default value for the `return_timeout` parameter is zero seconds. Therefore, if you don't include the parameter, the 202 HTTP status code is always returned for a valid request.

Querying the Job object associated with an API request

The Job object returned in the HTTP response contains several properties. You can query the state property in a subsequent API call to determine if the request completed successfully. A Job object is always in one of the following states:

Non-terminal states

- Queued
- Running
- Paused

Terminal states

- Success
- Failure

General procedure for issuing an asynchronous request

You can use the following high-level procedure to complete an asynchronous API call. This example assumes the `return_timeout` parameter is not used, or that the time expires before the background job completes.

1. Issue an API call that is designed to be performed asynchronously.
2. Receive an HTTP response 202 indicating acceptance of a valid request.
3. Extract the identifier for the Job object from the response body.
4. Within a timed loop, perform the following in each cycle:
   a. Get the current state of the Job.
   b. If the Job is in a non-terminal state, perform loop again.
5. Stop when the Job reaches a terminal state (success, failure).

Related information

- Update cluster contact
- Get job instance

Object references and access

The resource instances or objects exposed through the ONTAP REST API can be referenced and accessed in several different ways.

Object access paths

At a high level, there are two path types when accessing an object:

- Primary

  The object is the primary or direct target of the API call.

- Foreign

  The object is not the primary reference of the API call, but rather is linked to from the primary object. It is therefore a foreign or downstream object and referenced through a field in the primary object.

Accessing an object using the UUID

Every object is assigned a unique identifier when it is created, which in most cases is a 128-bit UUID. The assigned UUID values are immutable and are used internally within ONTAP to access and manage the resources. Because of this, the UUID generally provides the fastest and most stable way to access objects.

For many of the resource types, a UUID value can be provided as part of the path key in the URL to access a specific object. For example, you can use the following to access a node instance: `` `/cluster/nodes/{uuid} ``

Accessing an object using an object property

In addition to a UUID, you can also access an object using an object property. In most cases, it is convenient to use the name property. For example, you can use the following query parameter in the URL string to access a node instance by its name: `/cluster/nodes?name=node_one`. In addition to a query parameter, a foreign

object can be accessed through a property in the primary object.

While you can use the name or other property to access an object instead of the UUID, there are several possible disadvantages:

- The name field is not immutable and can be changed. If the name of an object is changed before accessing an object, the wrong object will be returned or an object access error will fail.

  This issue can occur with a POST or PATCH method on a foreign object or with a GET method on a primary object.

- ONTAP must translate the name field into the corresponding UUID. This is a type of indirect access which can become a performance issue.

In particular, a performance degradation is possible when one or more of the following is true:

- GET method is used
- A large collection of objects is accessed
- A complex or elaborate query is used

Cluster versus SVM context

There are several REST endpoints that support both a cluster and SVM. When using one of these endpoints, you can indicate the context of the API call through the `scope=[svm|cluster]` value. Examples of endpoints supporting a dual context include IP interfaces and security roles.

The scope value has a default value base on the properties provided for each API call.

Using PATCH and DELETE on a collection of objects

Every REST endpoint supporting PATCH or DELETE on a resource instance also supports the same method on a collection of objects. The only requirement is that at least one field must be provided through a query parameter in the URL string. When issuing a PATCH or DELETE over a collection, this is equivalent to doing the following internally:

- Query-based GET to retrieve the collection
- Serial sequence of PATCH or DELETE calls on each object in the collection

The time out for the operation can be set by `return_timeout` with a default of 15 seconds. If not completed before the timeout, the response includes a link to the next object. You must reissue the same HTTP method using the next link to continue the operation.

Performance metrics for storage resources

ONTAP collects performance metrics about selected SVM storage objects and protocols, and reports this information through the REST API. You can use this data to monitor the performance of an ONTAP system.

For a given storage object or protocol, the performance data falls into three categories:

- IOPS

- Latency
- Throughput

Within each category, one or more of the following types of data is available:

- Read ®
- Write (W)
- Other (O)
- Total (T)

The following table summarizes the performance data available through the ONTAP REST API, including the release when it was added. Refer to the REST API online documentation page at your ONTAP system for more information.

Storage object or protocol	IOPS	Latency	Throughput	ONTAP release
Ethernet port	Not applicable	Not applicable	RWT	9.8
FC port	RWOT	RWOT	RWT	9.8
IP interface	Not applicable	Not applicable	RWT	9.8
FC interface	RWOT	RWOT	RWT	9.8
NVMe namespace	RWOT	RWOT	RWOT	9.8
Qtree statistics	Raw RWOT	Not applicable	Raw RWOT	9.8
Volume Flexcache	RWOT	RWOT	RWT	9.8
Node – process utilization	Process utilization as a numerical value	Process utilization as a numerical value	Process utilization as a numerical value	9.8
Cloud volume	RWOT	RWOT	Not appliable	9.7
LUN	RWOT	RWOT	RWOT	9.7
Aggregate	RWOT	RWOT	RWOT	9.7
SVM NFS protocol	RWOT	RWOT	RWT	9.7
SVM CIFS protocol	RWOT	RWOT	RWT	9.7
SVM FCP protocol	RWOT	RWOT	RWT	9.7
SVM iSCSI protocol	RWOT	RWOT	RWT	9.7
SVM NVMe protocol	RWOT	RWOT	RWT	9.7
Cluster	RWOT	RWOT	RWOT	9.6
Volumes	RWOT	RWOT	RWOT	9.6

RBAC security

Overview of RBAC security

ONTAP includes a robust and extensible role-based access control (RBAC) capability. You can assign each account a different role to control the user's access to the resources exposed through the REST API and CLI. The roles define different levels of administrative access for the various ONTAP users.

The ONTAP RBAC capability has continued to expand and was significantly enhanced with ONTAP 9.11.1 (and subsequent releases). See Summary of RBAC evolution and What's new with the ONTAP REST API for more information.

ONTAP roles

A role is a set of privileges that collectively define what actions the user can take. Each privilege identifies a specific access path and the associated access level. Roles are assigned to user accounts and applied by ONTAP when making access control decisions.

Types of roles

There are two types of roles. They were introduced and tailored to different environments as ONTAP has evolved.

There are advantages and disadvantages when using each type of role. See Comparing the role types for more information.

Type	Description
REST	The REST roles were introduced with ONTAP 9.6 and are generally applied to users accessing ONTAP through the REST API. Creating a REST role automatically creates a traditional *mapping* role.
Traditional	These are the legacy roles included prior to ONTAP 9.6. They were introduced for the ONTAP CLI environment and continue to be fundamental to RBAC security.

Scope

Every role has a scope or context within which it is defined and applied. The scope determines where and how a specific role is used.

ONTAP user accounts also have a similar scope that determines how a user is defined and used.

Scope	Description
Cluster	Roles with a cluster scope are defined at the ONTAP cluster level. They are associated with cluster-level user accounts.
SVM	Roles with an SVM scope are defined for a specific data SVM. They are assigned to user accounts in the same SVM.

Source of the role definitions

There are two ways an ONTAP role can be defined.

Role source	Description
Custom	The ONTAP administrator can create custom roles. These roles can be tailored to a specific environment and security requirements.
Built-in	While custom roles provide more flexibility, there is also a set of built-in roles available at both the cluster and SVM level. These roles are pre-defined and can be used for many common administrative tasks.

Role mapping and ONTAP processing

Depending on the ONTAP release you are using, all or nearly all the REST API calls map to one or more CLI commands. When you create a REST role, a traditional or legacy role is also created. This **mapped** traditional role is based on the corresponding CLI commands and cannot be manipulated or changed.

Reverse role mapping is not supported. That is, creating a traditional role does not create a corresponding REST role.

Summary of RBAC evolution

The traditional roles are included with all ONTAP 9 releases. The REST roles were introduced later and have evolved as described below.

ONTAP 9.6

The REST API was introduced with ONTAP 9.6. The REST roles were included with this release as well. Also, when you create a REST role, a corresponding traditional role is also created.

ONTAP 9.7 through 9.10.1

Each ONTAP release from 9.7 through 9.10.1 includes enhancements to the REST API. For example, additional REST endpoints have been added with each release. However, the creation and management of the two roles types remained separate. Also, ONTAP 9.10.1 added REST RBAC support for the snapshots REST endpoint `/api/storage/volumes/{vol.uuid}/snapshots` which is a resource-qualified endpoint.

ONTAP 9.11.1

The ability to configure and manage traditional roles using the REST API was added with this release. Additional access levels for the REST roles were also added.

Work with roles and users

After understanding the basic RBAC capabilities, you can get started working with the ONTAP roles and users.

See RBAC workflows for examples of how to create and use roles with the ONTAP REST API.

Administrative access

You can create and manage the ONTAP roles through the REST API or command line interface. The access details are described below.

REST API

There are several endpoints that can be used when working with RBAC roles and user accounts. The first four in the table are used to create and manage the roles. The last two are used to create and manage user accounts.

You can access the ONTAP online API reference documentation for more information including examples of how to use the API.

Endpoint	Description
`/security/roles`	This endpoint allows you to create a new REST role. And beginning with ONTAP 9.11.1 you can also create a traditional role. In this case, ONTAP determines the role type based on the input parameters. You can also retrieve a list of the defined roles.
`/security/roles/{owner.UUID}/{name}`	You can retrieve or delete a specific cluster or SVM scoped role. The UUID value identifies the SVM where the role is defined (cluster or data SVM). The name value is the name of the role.
`/security/roles/{owner.UUID}/{name}/privileges`	This endpoint allows you to configure the privileges for a specific role. The built-in roles can be retrieved but not updated. See the API reference documentation for your ONTAP release for more information.
`/security/roles/{owner.UUID}/{name}/privileges/[path]`	You can retrieve, modify, and delete the access level and optional query value for a specific privilege. See the API reference documentation for your ONTAP release for more information.
`/security/accounts`	This endpoint allows you to create a new cluster or SVM scoped user account. Several types of information must be included or subsequently added before the account is operational. You can also retrieve a list of the defined user accounts.
`/security/accounts/{owner.UUID}/{name}`	You can retrieve, modify, and delete a specific cluster or SVM scoped user account. The UUID value identifies the SVM where the user is defined (cluster or data SVM). The name value is the name of the account.

Command line interface

The relevant ONTAP CLI commands are described below. All commands are accessed at the cluster level through an administrator account.

Command	Description
`security login`	This is the directory containing the commands needed to create and manage a user login.
`security login rest-role`	This is the directory containing the commands needed to create and manage a REST role associated with a user login.
`security login role`	This is the directory containing the commands needed to create and manage a traditional role associated with a user login.

Role definitions

The REST and traditional roles are defined through a set of attributes.

Owner and scope

A role can be owned by the ONTAP cluster or a specific data SVM within the cluster. The owner also implicitly determines the scope of the role.

Unique name

Every role must have a unique name within its scope. The name of a cluster role must be unique at the ONTAP cluster level while SVM roles must be unique within the specific SVM.

The name of a new REST role must be unique among the REST roles as well as the traditional roles. This is because creating a REST role also results in a new traditional *mapping* role with the same name.

Set of privileges

Every role contains a set of one or more privileges. Each privilege identifies a specific resource or command and the associated access level.

Privileges

A role can contain one or more privileges. Each privilege definition is a tuple and establishes the level of access to a specific resource or operation.

Resource path

The resource path is identified as either a REST endpoint or CLI command/command directory path.

REST endpoint

An API endpoint identified the target resource for a REST role.

CLI command

A CLI command identifies the target for a traditional role. A command directory can also be specified, which will then include all the downstream commands in the ONTAP CLI hierarchy.

Access level

The access level defines the type of access the role has to the specific resource path or command. The access levels are identified through a set of pre-defined keywords. Three access levels were introduced with ONTAP 9.6. They can be used for both traditional and REST roles. In addition, three new access levels were added with ONTAP 9.11.1. These new access levels can only be used with REST roles.

The access levels follow the CRUD model. With REST, this is based on the primary HTTP methods (POST, GET, PATCH, DELETE). The corresponding CLI operations generally map to the REST operations (create, show, modify, delete).

Access level	REST primitives	Added	REST role only
none	n/a	9.6	No
readonly	GET	9.6	No

Access level	REST primitives	Added	REST role only
all	GET, POST, PATCH, DELETE	9.6	No
read_create	GET, POST	9.11.1	Yes
read_modify	GET, PATCH	9.11.1	Yes
read_create_modify	GET, POST, PATCH	9.11.1	Yes

Optional query

When creating a traditional role, you can optionally include a **query** value to identify the subset of applicable objects for the command or command directory.

Summary of the built-in roles

There are several pre-defined roles included with ONTAP that you can use at either the cluster or SVM level.

Cluster scoped roles

There are several built-in roles available at the cluster scope.

See Predefined roles for cluster administrators for more information.

Role	Description
admin	Administrators with this role have unrestricted rights and can do anything in the ONTAP system. They can configure all cluster-level and SVM-level resources.
autosupport	This is a special role tailored for the AutoSupport account.
backup	This Special role for backup software that needs to back up the system.
snaplock	This is a special role tailored for the SnapLock account.
readonly	Administrators with this role can view everything at the cluster level but can’t make any changes.
none	No administrative capabilities are provided.

SVM scoped roles

There are several built-in roles available at the SVM scope. The **vsadmin** provides access to the most general and powerful capabilities. There are several additional roles tailored to specific administrative tasks, including:

- vsadmin-volume
- vsadmin-protocol
- vsadmin-backup
- vsadmin-snaplock
- vsadmin-readonly

See Predefined roles for SVM administrators for more information.

Comparing the role types

Before selecting a **REST** role or **traditional** role, you should be aware of the differences. Some of the ways the two role types can be compared are described below.

For more advanced or complex RBAC use cases, you should normally use a traditional role.

How the user accesses ONTAP

Before creating a role, it is important to know how the user will access the ONTAP system. Based on this a role type can be determined.

Access	Suggested type
REST API only	The REST role is designed to be used with the REST API.
REST API and CLI	You can define a REST role which also creates a corresponding traditional role.
CLI only	You can create a traditional role.

Precision of the access path

The access path defined for a REST role is based on a REST endpoint. The access path for a traditional role is based on a CLI command or command directory. In addition, you can include an optional query parameter with a traditional role to further restrict access based on the command parameter values.

Summary of the REST resources

Overview of the resource categories

The resources available through the ONTAP REST API are organized in categories. Each of the resource categories includes a brief description along with additional usage considerations where appropriate.

The REST resources described in the summary are based on the latest version of the product. If you need a more detailed understanding of the changes made in previous releases, see What's new with the ONTAP REST API as well as the ONTAP Release Notes.

For many of the REST endpoints, you can include a UUID key as part of the path string to access a specific object instance. However, in many cases you can also access objects using a property value on a query parameter.

Related information

- API reference

Application

You can use these API calls to manage the ONTAP application resources.

Application snapshots

Applications support snapshot copies, which can be created or restored at any time. This resource type was introduced with ONTAP 9.6.

Applications

The ONTAP applications are arranged based on type, including: templates, applications, components, and Snapshot copies. This resource type was introduced with ONTAP 9.6.

Consistency groups

A consistency group is a set of volumes that are grouped together when performing certain operations such as a snapshot. This feature extends the same crash consistency and data integrity implicit with single-volume operations across a set of volumes. This resource type was introduced with ONTAP 9.10 and updated with 9.12. An endpoint to retrieve metric performance and capacity data was added with ONTAP 9.13.

Consistency groups snapshots

You can use these endpoints to copy, create, inventory, and restore SnapShots for a consistency group. This resource type was introduced with ONTAP 9.10.

Cloud

You can use these API calls to manage connections to object storage resources in the cloud.

Targets

A target represents an object storage resource in the cloud. Each target includes the configuration information needed to connect to the storage resource. This resource type was introduced with ONTAP 9.6.

Cluster

You can use these API calls to manage ONTAP clusters and the related resources.

Capacity pools

The capacity pools licensing model allows you to license storage capacity for each cluster node from a shared pool. This resource type is new with ONTAP 9.8.

Chassis

The chassis is the hardware framework supporting a cluster. This resource type was introduced with ONTAP 9.6.

Clusters

An ONTAP cluster contains one or more nodes and the related configuration settings which define the storage system. This resource type was introduced with ONTAP 9.6.

Counter tables

Various statistical information about ONTAP is captured by the Counter Manager subsystem. You can access this information to assess system performance. This resource type was introduced with ONTAP 9.11.

Firmware

You can retrieve a history of the firmware update requests. This resource type is new with ONTAP 9.8.

Jobs

Asynchronous REST API requests are performed using a background task anchored by a job. This resource type was introduced with ONTAP 9.6.

License instance

Each license can be managed as a separate package. This resource type was introduced with ONTAP 9.6.

License managers

You can manage configuration and other information related to each license manager instance associated with an ONTAP cluster. This resource type is new with ONTAP 9.8.

Licenses

The licenses allow you to implement specific ONTAP features and functionality. This resource type was introduced with ONTAP 9.6.

Mediators

You can manage the mediator associated with MetroCluster, including adding or removing the mediator instance. This resource type is new with ONTAP 9.8.

MetroCluster

You can create and manage a MetroCluster deployment, including executing switchover or switchback operations. This resource type is new with ONTAP 9.8 and updated with 9.11.

MetroCluster diagnostics

You can perform a diagnostic operation on a MetroCluster deployment and retrieve the results. This resource type is new with ONTAP 9.8.

MetroCluster DR groups

You can perform operations related to the MetroCluster DR groups. This resource type is new with ONTAP 9.8.

MetroCluster interconnects

You can retrieve the MetroCluster interconnect status. This resource type is new with ONTAP 9.8.

MetroCluster nodes

You can retrieve the status of the individual nodes in a MetroCluster deployment. This resource type is new with ONTAP 9.8.

MetroCluster operations

You can retrieve a list of the recently executed operations for a MetroCluster configuration. This resource type is new with ONTAP 9.8.

MetroCluster SVMs

You can retrieve information about all the SVM pairs in a MetroCluster configuration. This resource type was introduced with ONTAP 9.11.1.

Nodes

ONTAP clusters are composed of one or more nodes. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.8.

NTP keys

The Network Time Protocol (NTP) can be configured to use shared private keys between ONTAP and trusted external NTP time servers. This resource type was introduced with ONTAP 9.7.

NTP server

You can use these API calls to configure the ONTAP Network Time Protocol settings, including the external NTP servers and keys. This resource type was introduced with ONTAP 9.7.

Peers

The peer objects represent endpoints and support the cluster peering relationships. This resource type was introduced with ONTAP 9.6.

Performance counters

Previous ONTAP releases have maintained statistical information about the operational characteristics of the system. With the 9.11.1 release, the information has been enhanced and is now available through the REST API. This feature brings the ONTAP REST API closer to parity with the Data ONTAP API (ONTAPI or ZAPI). This resource type was introduced with ONTAP 9.11.

Resource tags

You can use tags to group REST API resources. You might do this to associate related resources within a specific project or organizational group. Using tags can help to organize and track resources more effectively. This resource type was introduced with ONTAP 9.13.

Schedules

Schedules can be used to automate the perform of tasks. This resource type was introduced with ONTAP 9.6.

Sensors

You can use these endpoints to retrieve details about all the platform environment sensors. This resource type was introduced with ONTAP 9.11.

Software

An ONTAP cluster includes the cluster software profile, software packages collection, and software history collection. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.8.

Web

You can use these endpoints to update the web services configurations and to retrieve the current configuration. This resource type was introduced with ONTAP 9.10.

Name services

You can use these API calls to manage the name services supported by ONTAP.

Cache

ONTAP name services supports caching which improves performance and resiliency. Configuration of the name services cache can now be access through the REST API. Settings can be applied at multiple levels including: hosts, unix-users, unix-groups, and netgroups. This resource type was introduced with ONTAP 9.11.

DDNS

You can display the Dynamic DNS (DDNS) information and manage the DDNS subsystem. This resource type is new with ONTAP 9.8.

DNS

DNS supports the integration of the ONTAP cluster in your network. This resource type was introduced with ONTAP 9.6 and enhanced with ONTAP 9.13.

Host record

These endpoints allow you to displays the IP address of a specified hostname as well as the hostname for an IP address. This resource type was introduced with ONTAP 9.10.

LDAP

LDAP servers can be used to maintain user information. This resource type was introduced with ONTAP 9.6.

LDAP schemas

You can create, modify, and list the LDAP schemas used by ONTAP. There are four default schemas included. This resource type was introduced with ONTAP 9.11.

Local hosts

You can use these endpoints to display and manage the local mappings for hostnames. This resource type was introduced with ONTAP 9.10.

Name mappings

Name mappings allow you to map identities from one name domain to another. For example, you can map identities from CIFS to UNIX, Kerberos to UNIX, and UNIX to CIFS. This resource type was introduced with ONTAP 9.6.

Netgroup files

You can retrieve netgroup file details and delete a file for an SVM. This resource type was introduced with ONTAP 9.11.

NIS

NIS servers can be used to authenticate users and client workstations. This resource type was introduced with ONTAP 9.6.

UNIX users and groups

Local UNIX users and groups have been a part of previous ONTAP releases. However, support has now been added to the REST API allowing you to display and manage the users and groups. These REST resource types were introduced with ONTAP 9.9 and significantly enhanced with ONTAP 9.10.

NAS

You can use these API calls to manage the CIFS and NFS settings for the cluster and SVMs.

Active Directory

You can manage the Active Directory accounts defined for an ONTAP cluster. This includes creating new accounts as well as displaying, updating, and deleting accounts. This support was added with ONTAP 9.12.

Audit

Certain CIFS and NFS events can be logged for the SVMs, which can help to improve security. This resource type was introduced with ONTAP 9.6.

Audit log redirect

You can redirect NAS auditing events to a specific SVM. This resource type is new with ONTAP 9.8.

CIFS connections

You can retrieve a list of the established CIFS connections. This resource type was introduced with ONTAP 9.11.1.

CIFS domains

Support for CIFS domains has been added at the cluster and SVM level with several categories of endpoints. You can retrieve the domain configuration as well as create and remove preferred domain controllers. This resource type was introduced with ONTAP 9.10 and enhanced with ONTAP 9.13.

CIFS group policies

Endpoints has been added to support the creation and management of CIFS group policies. The configuration information is available and administered through group policy objects that are applied to all or specific SVMs. This support was added with ONTAP 9.12.

CIFS home directory search paths

Home directories for SMB users on a CIFS server can be created without creating an individual SMB share for each user. The home directory search path is a set of absolute paths from the root of an SVM. This resource type was introduced with ONTAP 9.6.

CIFS local groups

The CIFS server can use local groups for authorization when determining share, file, and directory access rights. This resource type was introduced with ONTAP 9.9 and significantly expanded with ONTAP 9.10.

CIFS NetBIOS

You can display information about the NetBIOS connections for the cluster. Details include the IP addresses and registered NetBIOS names. This information can help you to troubleshoot name resolution issues. This resource type was introduced with ONTAP 9.11.1.

CIFS services

The core configuration of the CIFS server. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.7 and 9.15.

CIFS session files

You can retrieve a list of open files for the CIFS sessions based on several filtering options. This resource type was introduced with ONTAP 9.11.1.

CIFS sessions

You can use this API to retrieve detailed information about a CIFS session. This resource type was introduced with the ONTAP 9.8 REST API and enhanced with ONTAP 9.9.

CIFS shadow copies

Microsoft Remote Volume Shadow Copy Services is an extension of the existing Microsoft VSS functionality. It extends the VSS capability to support shadow copying of SMB shares. This feature is now available through the ONTAP REST API. This resource type was introduced with ONTAP 9.11.1.

CIFS shares

The SMB shares defined at a CIFS server. This resource type was introduced with ONTAP 9.6.

CIFS shares ACLs

The access control lists (ACLs) controlling access to folders and files on the CIFS shares. This resource type was introduced with ONTAP 9.6.

CIFS UNIX symlink mapping

Both CIFS and UNIX clients can access the same datastore. When UNIX clients create symbolic links, these mappings provide a reference to another file or folder to support the CIFS clients. This resource type was introduced with ONTAP 9.6.

CIFS user and group bulk import

You can use the new REST API endpoints to perform a bulk import of the CIFS local users, groups, and group membership information as well as monitor the status of the request. This resource type was introduced with ONTAP 9.11.1.

File access tracing

You can use these API calls to trace access to specific files. This resource type is new with ONTAP 9.8.

File security permissions

You can use these API calls displays the effective permission granted to Windows or Unix user for a specific file or folder. You can also manage NTFS file security and audit policies. This resource type was introduced with the ONTAP 9.8 REST API and significantly enhanced with ONTAP 9.9.

FPolicy

FPolicy is a file access notification framework used to monitor and manage file access events on the SVMs. This resource type was introduced with ONTAP 9.6.

FPolicy connections

These endpoints allow you to display and update connection status information for external FPolicy servers. This resource type was introduced with ONTAP 9.10.

FPolicy engines

The FPolicy engines allow you to identify the external servers that receive the file access notifications. This resource type was introduced with ONTAP 9.6.

FPolicy events

The configuration identifying how file access is monitored and what events are generated. This resource type was introduced with ONTAP 9.6.

FPolicy persistent store

You can configure and administer a persistent store for the ONTAP FPolicy configuration and events. Each SVM can have one persistent store which is shared for the multiple policies within the SVM. This resource type was introduced with ONTAP 9.14.

FPolicy policies

A container for elements of the FPolicy framework, including FPolicy engines and events. This resource type was introduced with ONTAP 9.6.

Locks

A lock is a synchronization mechanism for enforcing limits on concurrent access to files where many clients are accessing the same file simultaneously. You can use these endpoints to retrieve and delete locks. This resource type was introduced with ONTAP 9.10.

NFS connected client maps

The NFS map information for the connected clients is available through the new endpoint. You can retrieve details about the node, SVM, and IP addresses. This resource type was introduced with ONTAP 9.11.1.

NFS connected clients

You can display a list of connected clients with the details of their connection. This resource type was introduced with ONTAP 9.7.

NFS export policies

The policies including rules that describe the NFS exports. This resource type was introduced with ONTAP 9.6.

NFS Kerberos interfaces

The configuration settings for an interface to Kerberos. This resource type was introduced with ONTAP 9.6.

NFS Kerberos realms

The configuration settings for Kerberos realms. This resource type was introduced with ONTAP 9.6.

NFS over TLS

This resource allows you to retrieve and update the interface configuration when using NFS over TLS. This resource type was introduced with ONTAP 9.15.

NFS services

The core configuration of the NFS server. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.7.

Object store

Auditing of the S3 events is a security improvement allowing you to track and log certain S3 events. An S3 audit event selector can be set on a per SVM per bucket basis. This resource type was introduced with ONTAP 9.10.

Vscan

A security feature to protect your data from viruses and other malicious code. This resource type was introduced with ONTAP 9.6.

Vscan on-access policies

The Vscan policies allowing files objects to be actively scanned when accessed by a client. This resource type was introduced with ONTAP 9.6.

Vscan on-demand policies

The Vscan policies allowing files objects to be immediately scanned on demand or according to a set schedule. This resource type was introduced with ONTAP 9.6.

Vscan scanner pools

A set of attributes used to manage the connection between ONTAP and an external virus-scanning server. This resource type was introduced with ONTAP 9.6.

Vscan server status

The status of the external virus-scanning server. This resource type was introduced with ONTAP 9.6.

NDMP

You can use these API calls to manage the NDMP services.

NDMP mode

The NDMP operational mode can be SVM scope or node scope. This resource type was introduced with ONTAP 9.7.

NDMP nodes

You can manage the NDMP configuration of the nodes. This resource type was introduced with ONTAP 9.7.

NDMP sessions

You can retrieve and delete NDMP session details for a specific SVM or node. This resource type was introduced with ONTAP 9.7.

NDMP SVMs

You can manage the NDMP configuration of the SVMs. This resource type was introduced with ONTAP 9.7.

NDMP SVM user passwords

You can generate and retrieve passwords for a specific NDMP user within the SVM content. This resource type was introduced with the ONTAP 9.8 REST API and enhanced with ONTAP 9.9.

Networking

You can use these API calls to manage the physical and logical networking resources used with the cluster.

BGP peer groups

You can create and administer Border Gateway Protocol peer groups. This resource type was introduced with ONTAP 9.7.

Ethernet broadcast domains

An Ethernet broadcast domain is a set of physical ports that appear to be part of the same physical network. All the ports receive a packet when broadcast from one of the ports in the domain. Each broadcast domain is part of an IPspace. This resource type was introduced with ONTAP 9.6.

Ethernet ports

An Ethernet port is a physical or virtual networking endpoint. The ports can be combined into a Link Aggregate Group (LAG) or separated using a Virtual LAN (VLAN). This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.8.

Ethernet switch ports

You can retrieve the port information for an Ethernet switch. This resource type is new with ONTAP 9.8.

Ethernet switches

You can retrieve or modify the configuration for Ethernet switches used for the ONTAP cluster or storage network. This resource type is new with ONTAP 9.8 and updated with 9.11.

Fibre Channel fabrics

You can use the Fibre Channel (FC) fabric REST API endpoints to retrieve information about the FC network. This includes the connections between the ONTAP cluster and the FC fabric, the switches comprising the fabric, and the zones of the active zoneset. This resource type was introduced with ONTAP 9.11.

Fibre Channel interfaces

A Fibre Channel interface is a logical endpoint associated with an SVM. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.8. Support for retrieving performance metrics data was added with ONTAP 9.14.

Fibre Channel ports

A Fibre Channel port is a physical adapter on an ONTAP node used to connect to the Fibre Channel network. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.8. Support for retrieving performance metrics data was added with ONTAP 9.14.

HTTP proxy

You can configure an HTTP proxy for either an SVM or a cluster IPSpace. This resource type was introduced with ONTAP 9.7.

IP interfaces

A logical interface (LIF) is an IP address with additional configuration attributes. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.8.

IP routes

A routing table is a collection of IP routes used to forward traffic to its destination. This resource type was introduced with ONTAP 9.6.

IP service policies

The IP service polices define the services available at a specific LIF. Service policies can be configured within the context of an SVM or IPspace. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.8.

IP subnets

The ONTAP networking capability has been expanded to support IP subnets. The REST API provides access to the configuration and management of the IP subnets within an ONTAP cluster. This resource type was introduced with ONTAP 9.11.

IPspaces

An IPspace creates a networking space to support one or more SVMs. The IPspaces can be isolated from each other, providing security and privacy. This resource type was introduced with ONTAP 9.6.

NVMe

You can use these API calls to manage resources supporting non-volatile memory express (NVMe).

Fibre Channel logins

Fibre Channel logins represent connections formed by Fibre Channel initiators logged in to ONTAP. This resource type was introduced with ONTAP 9.6.

Namespaces

An NVMe namespace is a collection of addressable logical blocks presented to hosts connected to the SVM using the NVMe over Fabrics protocol. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.8. Support for retrieving performance metrics data was added with ONTAP 9.14.

NVMe interfaces

NVMe interfaces are the network interfaces configured to support the NVMe over Fabrics (NVMe-oF) protocol. This resource type was introduced with ONTAP 9.6.

NVMe services

An NVMe service defines the properties of the NVMe controller target for an SVM. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.7. Support for retrieving performance metrics data was added with ONTAP 9.14.

NVMe subsystem controllers

The NVMe subsystem controllers represent dynamic connections between hosts and a storage solution. This resource type was introduced with ONTAP 9.6.

NVMe subsystem maps

An NVMe subsystem map is an association of an NVMe namespace with an NVMe subsystem. This resource type was introduced with ONTAP 9.6.

NVMe subsystems

An NVMe subsystem maintains configuration state and namespace access control for a set of NVMe-connected hosts. This resource type was introduced with ONTAP 9.6.

Object store

You can use these API calls to access S3-based object storage.

Buckets

A bucket is a container of objects and is structured using an object name space. Each S3 object server can have multiple buckets. This resource type was introduced with ONTAP 9.7 and updated with ONTAP 9.8.

Services

You can create and manage the ONTAP S3 configuration, including servers and bucket configurations. This resource type was introduced with ONTAP 9.7.

Service buckets

A bucket is a container of objects and is structured using an object name space. You can manage the buckets for a specific S3 server. This resource type was introduced with ONTAP 9.7.

S3 bucket rules

The S3 buckets can include a rule definition. Each rule is a list objects and defines the set of actions to be performed on an object within the bucket. This resource type was introduced with ONTAP 9.13.

S3 groups

You can create groups of S3 users and manage access control at the group level. This resource type is new with ONTAP 9.8.

S3 policies

You can create an S3 policy and associate it with a resource to define various permissions. This resource type is new with ONTAP 9.8.

Users

The S3 user accounts are maintained at the S3 server. User accounts are based on a pair of keys and associated with the buckets they control. This resource type was introduced with ONTAP 9.7.

SAN

You can use these API calls to manage storage area networking (SAN) resources.

Fibre Channel logins

Fibre Channel logins represent connections formed by Fibre Channel initiators that have logged in to ONTAP. This resource type was introduced with ONTAP 9.6.

Fibre Channel Protocol services

A Fibre Channel Protocol (FCP) service defines the properties of a Fibre Channel target for an SVM. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.7. Support for retrieving performance metrics data was added with ONTAP 9.14.

Fibre Channel WWPN aliases

A world wide port name (WWPN) is a 64-bit value uniquely identifying a Fibre Channel port. This resource type was introduced with ONTAP 9.6.

igroups

An initiator group (igroup) is a collection of Fibre Channel WWPNs (world wide port names), and iSCSI IQNs (qualified names), and iSCSI EUIs (extended unique identifiers) that identify host initiators. This resource type

was originally introduced with ONTAP 9.6.

Nested igroups is a new feature with ONTAP 9.9 and support has also been added to the REST API. This REST resource type was introduced with ONTAP 9.9.

Initiators

An initiator is a Fibre Channel (FC) world wide port name (WWPN), an iSCSI Qualified Name (IQN), or an iSCSI EUI (Extended Unique Identifier) that identifies a host endpoint. You can retrieve initiators for the cluster or a specific SVM. This resource type was introduced with ONTAP 9.14.

iSCSI credentials

The iSCSI credentials object contains authentication credentials which are used by an initiator and ONTAP. This resource type was introduced with ONTAP 9.6.

iSCSI services

An iSCSI service defines the properties of the iSCSI target for an SVM. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.7. Support for retrieving performance metrics data was added with ONTAP 9.14.

iSCSI sessions

An iSCSI session is one or more TCP connections that link an iSCSI initiator with an iSCSI target. This resource type was introduced with ONTAP 9.6.

LUN attributes

LUN attributes are caller-defined name/value pairs that can be optionally stored with a LUN. Attributes are available to save small amounts of application-specific metadata and are not interpreted by ONTAP. The endpoints allow you to create, update, delete, and discover attributes for a LUN. This resource type was introduced with ONTAP 9.10.

LUN maps

A LUN map is an association between a LUN and an initiator group. This resource type was introduced with ONTAP 9.6.

LUN maps reporting nodes

The reporting nodes are the cluster nodes from which network paths to a mapped LUN are advertised using the SAN protocols as part of the selective LUN map (SLM) feature of ONTAP. The new endpoints allow you to add, remove, and discover the reporting nodes of a LUN map. This resource type was introduced with ONTAP 9.10.

LUNs

A LUN is the logical representation of storage in a storage area network (SAN). This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.7. Support for retrieving performance metrics data was added with ONTAP 9.14.

Port sets

A port set is a collection of Fibre Channel or iSCSI network interfaces associated with the *portset* Storage VM. While this feature has existed with previous releases of ONTAP, support has now been added to the REST API. This REST resource type was introduced with ONTAP 9.9.

vVol bindings

A VMware virtual volume (vVol) binding is an association between a LUN of class `protocol_endpoint` and a LUN of class `vvol`. The vVol binding REST API allows you to create, delete, and discover vVol bindings. This resource type was introduced with ONTAP 9.10.

Security

You can use these API calls to manage the cluster and SVM security settings.

Accounts

There is a collection of user accounts for the cluster and SVMs. This resource type was introduced with ONTAP 9.6.

Accounts name

The configuration for a scoped user account. This resource type was introduced with ONTAP 9.6.

Active Directory proxy

You can administer the SVM account information at the Active Directory server. This resource type was introduced with ONTAP 9.7.

Anti-ransomware

ONTAP detects files potentially containing a ransomware threat. There are several categories of endpoints. You can retrieve a list of these suspect files as well as remove them from a volume. This resource type was introduced with ONTAP 9.10.1.

Audit

The settings which determine what is logged to the audit log files. This resource type was introduced with ONTAP 9.6.

Audit destinations

These settings control how audit log information is forwarded to remote systems or splunk servers. This resource type was introduced with ONTAP 9.6.

Audit messages

You can retrieve the audit log messages. This resource type was introduced with ONTAP 9.6.

AWS KMS

Amazon Web Services includes a key management service that provides secure storage for keys and other secrets. You can access this service through the REST API to allow ONTAP to securely store its encryption keys in the cloud. In addition, you can create and list the authentication keys used with NetApp Storage Encryption. This support is new with ONTAP 9.12.

Azure Key Vault

This set of API calls allows you to use the Azure Key Vault to store the ONTAP encryption keys. This resource type is new with ONTAP 9.8.

Certificates

The APIs calls can be used to install, display, and delete certificates used by ONTAP. This resource type was introduced with ONTAP 9.7.

Cisco Duo

Duo provides two-factor authentication for SSH logins. You can configure Duo to operate at the ONTAP cluster or SVM level. This resource type was introduced with ONTAP 9.14.

Cluster security

You can retrieve details of the cluster-wide security and update certain parameters. This resource type was introduced with ONTAP 9.7 and updated with ONTAP 9.8.

GCP KMS

This set of API calls allows you to use the Google Cloud Platform Key Management Service to store and manage the ONTAP encryption keys. This resource type was initially introduced with the ONTAP 9.8 REST API. However, this feature has been redesigned and so is considered to be new, with new resources types, in ONTAP 9.9.

IPSec

Internet Protocol Security (IPSec) is a suite of protocols providing security between two endpoints over an underlying IP network. This resource type is new with ONTAP 9.8.

IPSec CA certificates

You can add, remove, and retrieve IPSec CA certificates. This resource type is new with ONTAP 9.10.

IPSec policies

You can use this set of API calls to manage the policies in effect for an IPSec deployment. This resource type is new with ONTAP 9.8.

IPSec security associations

You can use this set of API calls to manage the security associations in effect for an IPSec deployment. This resource type is new with ONTAP 9.8.

Key manager configurations

These endpoints allow you to retrieve and update the configurations for key managers. This resource type is new with ONTAP 9.10.

Key managers

A key manager allows client modules within ONTAP to securely stored keys. This resource type was introduced with ONTAP 9.6 and updated for ONTAP 9.7. There was another update with ONTAP 9.12 to support authentication keys. A restore capability was added with ONTAP 9.13.

Key stores

A key store describes the type of a key manager. This resource type is new with ONTAP 9.10. Additional endpoints supporting enhanced control were added with ONTAP 9.14.

LDAP authentication

These API calls are used to retrieve and manage the cluster LDAP server configuration. This resource type was introduced with ONTAP 9.6.

Login messages

Used to display and manage the login messages used by ONTAP. This resource type was introduced with ONTAP 9.6.

Multiple administrator verification

The multiple administrator verification feature provides a flexible authorization framework for protecting access to ONTAP commands or operations. There are seventeen new endpoints that support defining, requesting, and approving access in the following areas:

- Rules
- Requests
- Approval groups

Providing the option for multiple administrators to approve access improves the security of your ONTAP and IT environments. These resource types were introduced with ONTAP 9.11.

NIS authentication

These settings are used to retrieve and manage the cluster NIS server configuration. This resource type was introduced with ONTAP 9.6.

OAuth 2.0

Open Authorization (OAuth 2.0) is a token-based framework that can be used to restrict access to your ONTAP storage resources. You can use it with clients that access ONTAP through the REST API. Configuration can be performed with any of the ONTAP administrative interfaces including the REST API. This resource type was introduced with ONTAP 9.14.

Password authentication

This includes the API call used to change the password for a user account. This resource type was introduced with ONTAP 9.6.

Privileges for a role instance

Manage the privileges for a specific role. This resource type was introduced with ONTAP 9.6.

Public key authentication

You can use these API calls to configure the public keys for user accounts. This resource type was introduced with ONTAP 9.7.

Roles

The roles provide a way to assign privileges to user accounts. This resource type was introduced with ONTAP 9.6.

Roles instance

Specific instance of a role. This resource type was introduced with ONTAP 9.6.

SAML service provider

You can display and manage the configuration for the SAML service provider. This resource type was introduced with ONTAP 9.6.

SSH

These calls allow you to set the SSH configuration. This resource type was introduced with ONTAP 9.7.

SSH SVMs

These endpoints allow you to retrieve the SSH security configuration for all SVMs. This resource type was introduced with ONTAP 9.10.

TOTPS

You can use the REST API to configure time-based one-time password (TOTP) profiles for accounts that sign in and access ONTAP using SSH. This resource type was introduced with ONTAP 9.13.

SnapLock

You can use these API calls to administer the ONTAP SnapLock feature.

Log

The SnapLock log structure is based on directories and files on a specific volume which contain the log

records. Log files are filled and archived based on the maximum log size. This resource type was introduced with ONTAP 9.7.

Compliance clock

The compliance clock determines the expiration time of the SnapLock objects. The clock must be initialized outside of the REST API and cannot be changed. This resource type was introduced with ONTAP 9.7.

Event retention

You can use the SnapLock Event Based Retention (EBR) feature to define how long a file is retained after the occurrence of an event. This resource type was introduced with ONTAP 9.7.

File retention and privileged delete

You can manage the retention time of a file created by SnapLock. If needed, you can also delete unexpired WORM files on a SnapLock enterprise volume. This resource type was introduced with ONTAP 9.7.

The only built-in role with authority to execute the delete operation is vsadmin-snaplock.

File fingerprint

You can view and manage the core information describing files and volumes, such as type and expiration date. This resource type was introduced with ONTAP 9.7.

Legal hold

You can use these API calls to manage files that are part of a litigation process. This resource type was introduced with ONTAP 9.7.

SnapMirror

You can use these API calls to manage the SnapMirror data protection technology.

Policies

The SnapMirror policies are applied to relationships, and control the configuration attributes and behavior of each relationship. This resource type was introduced with ONTAP 9.6.

Relationships

Both asynchronous and synchronous relationships establish the connectivity needed transfer data. This resource type was introduced with ONTAP 9.6.

Relationships transfers

You can manage the SnapMirror transfers over existing SnapMirror relationships. This resource type was introduced with ONTAP 9.6.

Storage

You can use these API calls to manage the physical and logical storage.

Aggregate metrics

You can retrieve historical metrics data for a specific aggregate. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.7.

Aggregate plexes

A physical copy of the WAFL storage within an aggregate. This resource type was introduced with ONTAP 9.6.

Aggregates

An aggregate consists of one or more RAID groups. This resource type was introduced with ONTAP 9.6.

Bridges

You can retrieve the bridges in a cluster. This resource type was introduced with ONTAP 9.9.

Disks

The physical disks in the cluster. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.7 and 9.8.

File clone

You can use these endpoints to create file clones, retrieve split status, and manage split loads. The file cloning endpoint resources were first introduced with ONTAP 9.6 and expanded with ONTAP 9.8. They were significantly expanded again with ONTAP 9.10.

File moves

You can use these REST API endpoints to move a file between two FlexVol volumes or within a FlexGroup volume. After the request is accepted you can monitor the progress and status. This resource type was introduced with ONTAP 9.11.1.

FlexCache

This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.8.

FlexCache origins

FlexCache is a persistent cache of an origin volume. This resource type was originally introduced with ONTAP 9.6. Support has been enhanced with the ONTAP 9.9 REST API to support modification through the HTTP PATCH method.

Monitored files

You can designate specific files for additional monitoring. This resource type is new with ONTAP 9.8.

Pools

You can create a shared storage pool as well as retrieve the storage pools in a cluster. This resource type was introduced with ONTAP 9.11.1.

Ports

Storage ports of the cluster. This resource type was introduced with ONTAP 9.6 and enhanced with ONTAP 9.11.1.

QOS policies

Quality of service policy configuration. This resource type was introduced with ONTAP 9.6.

QOS options

Endpoints have been introduced to allow you to retrieve and set QOS options for the cluster. For example, you can reserve a percentage of available system processing resources for background tasks. This resource type was introduced with ONTAP 9.14.

QOS workloads

A QOS workload represents a storage object tracked by QOS. You can retrieve the QOS workflows. This resource type was introduced with ONTAP 9.10.

Qtrees

You can use these API calls to management Qtrees, a type of logically divided file system. This resource type was introduced with ONTAP 9.6.

Quota reports

Report on quotas, which is a technique for restricting or tracking files or space usage. This resource type was introduced with ONTAP 9.6.

Quota rules

The rules used to enforce the quotas. This resource type was introduced with ONTAP 9.6 and updated with ONTAP 9.7.

Shelves

Shelves in the cluster. This resource type was introduced with ONTAP 9.6.

Snapshot policies

Snapshots are created based on policies. This resource type was introduced with ONTAP 9.6.

Snapshot schedules

You can control the snapshot schedules. This resource type is newly redesigned with ONTAP 9.8.

Switches

You can retrieve the switches in a cluster. This resource type was introduced with ONTAP 9.9.

Tape devices

You can retrieve the tape devices in a cluster. This resource type was introduced with ONTAP 9.9.

Top metrics

The top metrics endpoints allow you to determine activity for a volume filtered by a specific metric. Filtering can be done based on clients, directories, files, and users. This resource type was introduced with ONTAP 9.10.

Volume efficiency policies

You can use these API calls to configure the efficiencies applied to an entire volume. This resource type is new with ONTAP 9.8.

Volumes

Logical containers are used to serve data to clients. This resource type was originally introduced with ONTAP 9.6 REST API. Many of the parameter values used with the API were significantly expanded with ONTAP 9.9 including those used with space management.

Volume files

You can retrieve a list of files and directories for a specific directory on a volume. This resource type was introduced with ONTAP 9.7 and updated with ONTAP 9.8.

Volumes Snapshots

Snapshots for a volume. This resource type was introduced with ONTAP 9.6.

Support

You can use these API calls to manage the ONTAP features used to support a cluster.

Application log

A standalone application can record EMS events and optionally generated AutoSupport packages at an

ONTAP system by issuing a POST request. This resource type was introduced with ONTAP 9.11.1

Automatic update

The automatic update feature keeps your ONTAP systems current by downloading and applying the latest software updates. There are several endpoint categories to support the feature, including status, configurations, and updates. These resource types were introduced with ONTAP 9.10.

AutoSupport

AutoSupport collects configuration and status details as well as errors, and reports the information to NetApp. This resource type was introduced with ONTAP 9.6.

AutoSupport messages

Each node maintains AutoSupport messages that can be generated and retrieved. This resource type was introduced with ONTAP 9.6.

Configuration backup

You can use these APIs to retrieve and update the current backup settings. This resource type was introduced with ONTAP 9.6.

Configuration backup operations

You can create, retrieve, and delete configuration backup files. This resource type was introduced with ONTAP 9.7.

Core dump

You can use these endpoints to retrieve and manage the memory core dumps generated by a cluster or node. This resource type was introduced with ONTAP 9.10.

EMS

The event management system (EMS) collects events and sends notifications to one or more destinations. This resource type was introduced with ONTAP 9.6.

EMS destinations

The EMS destinations determine how and where notifications are sent. This resource type was introduced with ONTAP 9.6.

EMS destinations instance

An EMS destination instance is defined by type and location. This resource type was introduced with ONTAP 9.6.

EMS events

This is a live collection of system events for the cluster. This resource type was introduced with ONTAP 9.6.

EMS filters

The EMS filters collectively identify the events that require additional processing. This resource type was introduced with ONTAP 9.6.

EMS filters instance

An EMS filter instance is a collection of rules that are applied to the events. This resource type was introduced with ONTAP 9.6.

EMS messages

Provides access to the EMS event catalog. This resource type was introduced with ONTAP 9.6.

EMS role configuration

The EMS support feature allows for the management of roles and the access control configuration assigned to the roles. This provides the ability to limit or filter the events and messages based on the role configuration. This resource type was introduced with ONTAP 9.13.

EMS rules for filter instance

A list of rules can be managed for a specific instance of an EMS filter. This resource type was introduced with ONTAP 9.6.

EMS rules instance for filter instance

An individual rule for a specific instance of an EMS filter. This resource type was introduced with ONTAP 9.6.

SNMP

You can enable and disable SNMP and trap operations for the cluster. This resource type was introduced with ONTAP 9.7.

SNMP trap host

An SNMP trap host is a system that is configured to receive SNMP traps from ONTAP. You can retrieve and define the hosts. This resource type was introduced with ONTAP 9.7.

SNMP trap host instance

You can manage specific SNMP trap hosts. This resource type was introduced with ONTAP 9.7.

SNMP users

You can define and administer SNMP users. This resource type was introduced with ONTAP 9.7.

SNMP users instance

You can administer a specific SNMP user where the engine ID is associated with the administrative SVM or a data SVM. This resource type was introduced with ONTAP 9.7.

SVM

You can use these API calls to manage storage virtual machines (SVMs).

Migrations

You can migrate an SVM from a source cluster to a destination cluster. The new endpoints provide complete control, including the ability to pause, resume, retrieve status, and abort a migration operation. This resource type was introduced with ONTAP 9.10.

Peer permissions

Peer permissions can be assigned which enable the SVM peering relationships. This resource type was introduced with ONTAP 9.6.

Peers

The peering relationships establish connectivity among the SVMs. This resource type was introduced with ONTAP 9.6.

SVMs

You can manage the SVMs that are bound to a cluster. This resource type was introduced with ONTAP 9.6.

Top metrics

You can access additional performance metrics data for a specific SVM instance. There are four lists available and each provides the top I/O activity for ONTAP FlexVol and FlexGroup volumes. The lists include:

- Clients
- Directories
- Files
- Users

These resource types were introduced with ONTAP 9.11.

Web

You can use these endpoints to update and retrieve the web services security configuration for each data SVM. This resource type was introduced with ONTAP 9.10.

Workflows

Prepare to use the workflows

You should be familiar with the structure and format of the workflows before using them with a live ONTAP deployment.

You should make sure that your ONTAP release supports all the API calls in the workflows you plan to use. See API reference for more information.

Introduction

A *workflow* is a sequence of one or more steps needed to accomplish a specific administrative task or goal. The ONTAP workflows include the core steps and parameters you need to accomplish each task. They provide a starting point for customizing your ONTAP automation environment.

Step types

Each step in an ONTAP workflow is one of the following types:

- REST API call (with details such as curl and JSON examples)
- Perform or invoke another ONTAP workflow
- Miscellaneous related task (such as making a configuration decision)

REST API calls

Most of the workflow steps are REST API calls. These steps use a common format which includes a curl example and other information. See the API reference for more details about the REST API calls.

Single-step workflows

A workflow can contain only one step. These *single-step workflows* are formatted slightly differently than workflows containing multiple steps. For example, the explicit step name is removed. The action or operation should be clear based on the workflow title.

Input variables

The workflows are designed to be as general as possible so they can be used in any ONTAP environment. With this in mind, the REST API calls use variables in the curl examples and other input. The REST API calls can then be easily adapted to different ONTAP environments.

Base URL format

You can access the ONTAP REST API directly through curl or a programming language. In this case, the base URL is different than the URL you use when accessing the ONTAP online documentation or System Manager.

When accessing the API directly, you need to append **api** to the domain or IP address. For example:

```
https://ontap.demo-example.com/api
```

See How to access the ONTAP REST API for more information.

Common input parameters

There are several input parameters commonly used with most of the REST API calls. These parameters are typically not described in the individual workflows. You should be familiar with the parameters. See Input variables controlling an API request for more information.

If additional parameters are needed for a specific REST API call, they are included in the section **Additional input parameters for the curl example** for each workflow.

Variable format

The ID values and other variables used with the workflow examples are opaque and can vary with each ONTAP cluster. To improve the readability of the examples, actual values are not used. Variables are used instead. This approach, based on a consistent format and set of reserved names, has several benefits including:

- The curl and JSON samples are more readable and easier to understand.
- Because all the keywords use the same format, you can quickly identify them.
- There is no security exposure because the values cannot be copied and reused.

The variables are formatted to be used in a Bash shell environment. Each variable begins with a dollar sign and is enclosed in double quotes as needed. This makes them recognizable to Bash. Upper case is consistently used for the names.

Here are some of the common variable keywords. This list is not exhaustive and additional variables are used as needed. Their meaning should be obvious based on the context.

Keyword	Type	Description
$FQDN_IP	URL	The fully qualified domain name or IP address of the ONTAP management LIF.
$CLUSTER_ID	Path	The UUIDv4 value identifying the ONTAP cluster where the API operations run.
$BASIC_AUTH	Header	The credentials string used for HTTP basic authentication.

JSON input examples

Some of the REST API calls, such as those using POST or PATCH, require JSON input in the body of the request. The JSON input examples are presented separately from the curl examples for clarity. You can use the JSON input examples with one of the techniques described below.

Save to local file

You can copy the JSON input example to a file and save it locally. The curl command refers to the file using the `--data` parameter with the value indicating the file name with a `@` prefix.

Paste into terminal after the curl example

First you need to copy and paste the curl example into a terminal shell. Then edit the example to completely remove the `--data` parameter at the end and replace it with the `--data-raw` parameter. Finally, copy and paste in the JSON example so that it follows the curl command with the updated parameter. You should use single quotes to wrap the JSON input example.

Authentication options

The primary authentication technique available for the REST API is HTTP basic authentication. Beginning with ONTAP 9.14, you also have the option of using the Open Authorization (OAuth 2.0) framework with token-based authentication and authorization.

HTTP basic authentication

When using basic authentication, the user credentials must be included with each HTTP request. There are two options for sending the credentials.

Construct the HTTP request header

You can manually construct the Authorization header and include it with the HTTP requests. This can be done when using a curl command in the CLI or a programming language with your automation code. The high-level steps include:

1. Concatenate the user and password values with a colon:

   ```
   admin:david123
   ```

2. Convert the entire string to base64:

   ```
   YWRtaW46ZGF2aWQxMjM=
   ```

3. Construct the request header:

   ```
   Authorization: Basic YWRtaW46ZGF2aWQxMjM=
   ```

The workflow curl examples include this header with the variable **$BASIC_AUTH** which you need to update before using.

Use a curl parameter

Another option when using curl is to remove the Authorization header and use the curl **user** parameter instead. For example:

```
--user username:password
```

You need to substitute the appropriate credentials for your environment. The credentials are not encoded in base64. When executing the curl command with this parameter, the string is encoded and the Authorization header is generated for you.

OAuth 2.0

When using OAuth 2.0, you need to request an access token from an external authorization server and include it with each HTTP request. The basic high-level steps are described below. Also see Overview of the ONTAP OAuth 2.0 implementation for more details about OAuth 2.0 and how to use it with ONTAP.

Prepare your ONTAP environment

Before using the REST API to access ONTAP, you need to prepare and configure the ONTAP environment. At a high level, the steps include:

- Identify the ONTAP protected resources and clients
- Review the existing ONTAP REST role and user definitions

- Install and configure the authorization server
- Design and configure the client authorization definitions
- Configure ONTAP and enable OAuth 2.0

Request an access token

With ONTAP and the authorization server defined and active, you can make a REST API call using an OAuth 2.0 token. The first step is to request an access token from the authorization server. This is done outside of ONTAP using one of several different techniques based on the server. ONTAP does not issue access tokens or perform redirection.

Construct the HTTP request header

After obtaining an access token, you can construct an Authorization header and include it with the HTTP requests. Regardless of whether you use curl or a programming language to access the REST API, you must include the header with every client request. You can construct the header as follows:

```
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSld …
```

Using the examples with Bash

If you use the workflow curl examples directly, you must update the variables they contain with values appropriate for your environment. You can manually edit the examples or rely on the Bash shell to do the substitution for you as described below.

> One advantage of using Bash is that you can set the variable values one time in a shell session instead of once per curl command.

Steps

1. Open the Bash shell provided with Linux or similar operating system.
2. Set the variable values included in the curl example you want to run. For example:

   ```
   CLUSTER_ID=ce559b75-4145-11ee-b51a-005056aee9fb
   ```

3. Copy the curl example from the workflow page and paste it into the shell terminal.
4. Press **ENTER** which will do the following:
   a. Substitute the variable values you set
   b. Execute the curl command

Cluster

Get cluster configuration

You can retrieve the configuration for an ONTAP cluster including specific fields. You might do this as part of assessing the state of the cluster or before updating the configuration.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/cluster

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
fields	Query	No	Select the values you want returned. Examples include `contact` and `version`.

Curl example: Retrieve the cluster contact information

This example illustrates how to retrieve a single field. To get the entire cluster object and configuration, you need to remove the `fields` query parameter.

```
curl --request GET \
--location "https://$FQDN_IP/api/cluster?fields=contact" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "contact": "support@company-demo.com"
}
```

Update cluster contact

You can update the contact information for a cluster. Because the request is processed asynchronously, you also need to determine if the associated background job completed successfully.

Step 1: Update the cluster contact information

You can issue an API call to update the cluster contact information.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
PATCH	/api/cluster

Processing type

Asynchronous

Curl example

```
curl --request PATCH \
--location "https://$FQDN_IP/api/cluster" \
--include \
--header "Content-Type: application/json" \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "contact": "support@company-demo.com"
}
```

JSON output example

A job object is returned. You should save the job identifier to use it in the next step.

```
{ "job": {
    "uuid": "d877f5bb-3aa7-11e9-b6c6-005056a78c89",
    "_links": {
      "self": {
        "href": "/api/cluster/jobs/d877f5bb-3aa7-11e9-b6c6-005056a78c89"
      }
    }
  }
}
```

Step 2: Retrieve the status of the job

Perform the workflow Get job instance and confirm the `state` value is `success`.

Step 3: Confirm the cluster contact information

Perform the workflow Get cluster configuration. You should set the `fields` query parameter to `contact`.

Get job instance

You can retrieve the instance of a specific ONTAP job. You would typically do this to determine if the job and associated operation completed successfully.

You need the UUID of the job object, which is typically provided after issuing an asynchronous request. Also review Asynchronous processing using the Job object before working with ONTAP internal jobs.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/cluster/jobs/{uuid}

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
$JOB_ID	Path	Yes	Needed to identify the job being requested.

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/cluster/jobs/$JOB_ID" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

The state value and other fields are included in the returned job object. The job in this example was run as part of updating an ONTAP cluster.


```
{
  "uuid": "d877f5bb-3aa7-11e9-b6c6-005056a78c89",
  "description": "PATCH /api/cluster",
  "state": "success",
  "message": "success",
  "code": 0,
  "_links": {
    "self": {
      "href": "/api/cluster/jobs/d877f5bb-3aa7-11e9-b6c6-005056a78c89"
    }
  }
}
```


NAS

File security permissions

Prepare to manage file security and audit policies

You can manage the permissions and audit policies for files available through the SVMs within an ONTAP cluster.

Overview

ONTAP uses System Access Control Lists (SACLs) and Discretionary Access Control Lists (DACLs) to assign permissions to file objects. Beginning with ONTAP 9.9.1, the REST API includes support for managing the SACL and DACL permissions. You can use the API to automate the administration of the file security permissions. In many cases you can use a single REST API call instead of multiple CLI commands or ONTAPI (ZAPI) calls.

For ONTAP releases prior to 9.9.1, you can automate the administration of the SACL and DACL permissions using the CLI passthrough feature. See Migration considerations and Using the private CLI passthrough with the ONTAP REST API for more information.

Several example workflows are available to illustrate how to manage the ONTAP file security services using the REST API. Before using the workflows and issuing any of the REST API calls, make sure to review Prepare to use the workflows.

If you use Python, also see the script file_security_permissions.py for examples of how to automate some of the file security activities.

ONTAP REST API versus ONTAP CLI commands

For many tasks, using the ONTAP REST API requires fewer calls than the equivalent ONTAP CLI commands or ONTAPI (ZAPI) calls. The table below includes a list of API calls and the equivalent the CLI commands needed for each task.

ONTAP REST API	ONTAP CLI
`GET /protocols/file-security/effective-permissions/`	`vserver security file-directory show-effective-permissions`
`POST /protocols/file-security/permissions/`	1. `vserver security file-directory ntfs create` 2. `vserver security file-directory ntfs dacl add` 3. `vserver security file-directory ntfs sacl add` 4. `vserver security file-directory policy create` 5. `vserver security file-directory policy task add` 6. `vserver security file-directory apply`
`PATCH /protocols/file-security/permissions/`	`vserver security file-directory ntfs modify`

ONTAP REST API	ONTAP CLI
`DELETE /protocols/file-security/permissions/`	1. `vserver security file-directory ntfs dacl remove` 2. `vserver security file-directory ntfs sacl remove`

Related information

- Python script illustrating file permissions
- Simplified management of file-security permissions with ONTAP REST APIs
- Using the private CLI passthrough with the ONTAP REST API

Get the effective permissions for a file

You can retrieve the current effective permissions for a specific file or folder.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/protocols/file-security/effective-permissions/{svm.uuid}/{path}

Processing type

Synchronous

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM containing the file.
$FILE_PATH	Path	Yes	This is the path to the file or folder.

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/protocols/file-security/effective-
permissions/$SVM_ID/$FILE_PATH" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example


```
{
  "svm": {
    "uuid": "cf5f271a-1beb-11ea-8fad-005056bb645e",
    "name": "vs1"
  },
  "user": "administrator",
  "type": "windows",
  "path": "/",
  "share": {
    "path": "/"
  },
  "file_permission": [
    "read",
    "write",
    "append",
    "read_ea",
    "write_ea",
    "execute",
    "delete_child",
    "read_attributes",
    "write_attributes",
    "delete",
    "read_control",
    "write_dac",
    "write_owner",
    "synchronize",
    "system_security"
  ],
  "share_permission": [
    "read",
    "read_ea",
    "execute",
    "read_attributes",
    "read_control",
    "synchronize"
  ]
}
```


Get the auditing information for a file

You can retrieve the auditing information for a specific file or folder.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/protocols/file-security/permissions/{svm.uuid}/{path}

Processing type

Synchronous

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM containing the file.
$FILE_PATH	Path	Yes	This is the path to the file or folder.

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/protocols/file-
security/permissions/$SVM_ID/$FILE_PATH" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "svm": {
    "uuid": "9479099d-5b9f-11eb-9c4e-0050568e8682",
    "name": "vs1"
  },
  "path": "/parent",
  "owner": "BUILTIN\\Administrators",
  "group": "BUILTIN\\Administrators",
  "control_flags": "0x8014",
  "acls": [
    {
      "user": "BUILTIN\\Administrators",
      "access": "access_allow",
      "apply_to": {
        "files": true,
        "sub_folders": true,
        "this_folder": true
      },
      "advanced_rights": {
        "append_data": true,
```

```
        "delete": true,
        "delete_child": true,
        "execute_file": true,
        "full_control": true,
        "read_attr": true,
        "read_data": true,
        "read_ea": true,
        "read_perm": true,
        "write_attr": true,
        "write_data": true,
        "write_ea": true,
        "write_owner": true,
        "synchronize": true,
        "write_perm": true
      },
      "access_control": "file_directory"
    },
    {
      "user": "BUILTIN\\Users",
      "access": "access_allow",
      "apply_to": {
        "files": true,
        "sub_folders": true,
        "this_folder": true
      },
      "advanced_rights": {
        "append_data": true,
        "delete": true,
        "delete_child": true,
        "execute_file": true,
        "full_control": true,
        "read_attr": true,
        "read_data": true,
        "read_ea": true,
        "read_perm": true,
        "write_attr": true,
        "write_data": true,
        "write_ea": true,
        "write_owner": true,
        "synchronize": true,
        "write_perm": true
      },
      "access_control": "file_directory"
    }
  ],
  "inode": 64,
```

```
  "security_style": "mixed",
  "effective_style": "ntfs",
  "dos_attributes": "10",
  "text_dos_attr": "----D---",
  "user_id": "0",
  "group_id": "0",
  "mode_bits": 777,
  "text_mode_bits": "rwxrwxrwx"
}
```

Apply new permissions to a file

You can apply a new security descriptor to a specific file or folder.

Step 1: Apply the new permissions

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/protocols/file-security/permissions/{svm.uuid}/{path}

Processing type

Asynchronous

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM containing the file.
$FILE_PATH	Path	Yes	This is the path to the file or folder.

Curl example

```
curl --request POST --location "https://$FQDN_IP/api/protocols/file-
security/permissions/$SVM_ID/$FILE_PATH?return_timeout=0" --include
--header "Accept */*" --header "Authorization: Basic $BASIC_AUTH" --data
'{ \"acls\": [ { \"access\": \"access_allow\", \"advanced_rights\": {
\"append_data\": true, \"delete\": true, \"delete_child\": true,
\"execute_file\": true, \"full_control\": true, \"read_attr\": true,
\"read_data\": true, \"read_ea\": true, \"read_perm\": true,
\"write_attr\": true, \"write_data\": true, \"write_ea\": true,
\"write_owner\": true, \"write_perm\": true }, \"apply_to\": { \"files\":
true, \"sub_folders\": true, \"this_folder\": true }, \"user\":
\"administrator\" } ], \"control_flags\": \"32788\", \"group\": \"S-1-5-
21-2233347455-2266964949-1780268902-69700\", \"ignore_paths\": [
\"/parent/child2\" ], \"owner\": \"S-1-5-21-2233347455-2266964949-
1780268902-69304\", \"propagation_mode\": \"propagate\"}'
```

JSON output example

```
{
  "job": {
    "uuid": "3015c294-5bbc-11eb-9c4e-0050568e8682",
    "_links": {
      "self": {
        "href": "/api/cluster/jobs/3015c294-5bbc-11eb-9c4e-0050568e8682"
      }
    }
  }
}
```

Step 2: Retrieve the status of the job

Perform the workflow Get job instance and confirm the `state` value is `success`.

Update security descriptor information

You can update a specific security descriptor to a specific file or folder, including the primary owner, group, or control flags.

Step 1: Update the security descriptor

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
PATCH	/api/protocols/file-security/permissions/{svm.uuid}/{path}

Processing type

Asynchronous

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM containing the file.
$FILE_PATH	Path	Yes	This is the path to the file or folder.

Curl example

```
curl --request POST --location "https://$FQDN_IP/api/protocols/file-
security/permissions/$SVM_ID/$FILE_PATH?return_timeout=0" --include
--header "Accept */*" --header "Authorization: Basic $BASIC_AUTH" --data
'{ \"control_flags\": \"32788\", \"group\": \"everyone\", \"owner\":
\"user1\"}'
```

JSON output example

```
{
  "job": {
    "uuid": "6f89e612-5bbd-11eb-9c4e-0050568e8682",
    "_links": {
      "self": {
        "href": "/api/cluster/jobs/6f89e612-5bbd-11eb-9c4e-0050568e8682"
      }
    }
  }
}
```

Step 2: Retrieve the status of the job

Perform the workflow Get job instance and confirm the `state` value is `success`.

Delete an access control entry

You can delete an existing Access Control Entry (ACE) from a specific file or folder. The change propagates to any child objects.

Step 1: Delete the ACE

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
DELETE	/api/protocols/file-security/permissions/{svm.uuid}/{path}

Processing type

Asynchronous

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM containing the file.
$FILE_PATH	Path	Yes	This is the path to the file or folder.

Curl example

```
curl --request DELETE --location "https://$FQDN_IP/api/protocols/file-
security/permissions/$SVM_ID/$FILE_PATH?return_timeout=0" --include
--header "Accept */*" --header "Authorization: Basic $BASIC_AUTH" --data
'{ \"access\": \"access_allow\", \"apply_to\": { \"files\": true,
\"sub_folders\": true, \"this_folder\": true }, \"ignore_paths\": [
\"/parent/child2\" ], \"propagation_mode\": \"propagate\"}'
```

JSON output example

```
{
  "job": {
    "uuid": "3015c294-5bbc-11eb-9c4e-0050568e8682",
    "_links": {
      "self": {
        "href": "/api/cluster/jobs/3015c294-5bbc-11eb-9c4e-0050568e8682"
      }
    }
  }
}
```

Step 2: Retrieve the status of the job

Perform the workflow Get job instance and confirm the `state` value is `success`.

Networking

List the IP interfaces

You can retrieve the IP LIFs assigned to the cluster and SVMs. You might do this to

confirm your network configuration or when planning to add another LIF.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/network/ip/interfaces

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
fields	Query	No	Return a limited list of the relevant configuration values.

Curl example: return all LIFs with the default configuration values

```
curl --request GET \
--location "https://$FQDN_IP/api/network/ip/interfaces" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

Curl example: Return all LIFs with four specific configuration values

```
curl --request GET \
--location
```

```
"https://$FQDN_IP/api/network/ip/interfaces?fields=name,scope,svm.name,ip.
address" \
3 --include \
4 --header "Accept: */*" \
5 --header "Authorization: Basic $BASIC_AUTH"
```

JSON output example


```
{
  "records": [
    {
      "uuid": "5ded9e38-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_mgmt1",
      "ip": {
        "address": "172.29.151.116"
      },
      "scope": "cluster",
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/5ded9e38-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "bb03c162-999e-11ee-acad-005056ae6bd8",
      "name": "cluster_mgmt",
      "ip": {
        "address": "172.29.186.156"
      },
      "scope": "cluster",
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/bb03c162-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "c5ffbd03-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data1",
      "ip": {
        "address": "172.29.186.150"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c5ffbd03-999e-11ee-acad-
005056ae6bd8"
```

```
        }
      }
    },
    {
      "uuid": "c6612abe-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data2",
      "ip": {
        "address": "172.29.186.151"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c6612abe-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "c6b21b94-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data3",
      "ip": {
        "address": "172.29.186.152"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c6b21b94-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "c7025322-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data4",
      "ip": {
        "address": "172.29.186.153"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
```

```
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c7025322-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "c752cc66-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data5",
      "ip": {
        "address": "172.29.186.154"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c752cc66-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "c7a03719-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data6",
      "ip": {
        "address": "172.29.186.155"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/c7a03719-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "ccd4c59c-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data4_inet6",
      "ip": {
```

```
        "address": "fd20:8b1e:b255:300f::ac5"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/ccd4c59c-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "d9144c30-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data6_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac7"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/d9144c30-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "d961c13b-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data1_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac2"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/d961c13b-999e-11ee-acad-
005056ae6bd8"
        }
      }
```

```
    },
    {
      "uuid": "d9ac8d6a-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data5_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac6"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/d9ac8d6a-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "d9fce1a3-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data2_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac3"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/d9fce1a3-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "da4995a0-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_data3_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac4"
      },
      "scope": "svm",
      "svm": {
        "name": "vs0"
      },
      "_links": {
```

```
        "self": {
          "href": "/api/network/ip/interfaces/da4995a0-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "da9e7afd-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_cluster_mgmt_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:300f::ac8"
      },
      "scope": "cluster",
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/da9e7afd-999e-11ee-acad-
005056ae6bd8"
        }
      }
    },
    {
      "uuid": "e6db58b4-999e-11ee-acad-005056ae6bd8",
      "name": "sti214-vsim-sr027o_mgmt1_inet6",
      "ip": {
        "address": "fd20:8b1e:b255:3008::1a0"
      },
      "scope": "cluster",
      "_links": {
        "self": {
          "href": "/api/network/ip/interfaces/e6db58b4-999e-11ee-acad-
005056ae6bd8"
        }
      }
    }
  ],
  "num_records": 16,
  "_links": {
    "self": {
      "href":
"/api/network/ip/interfaces?fields=name,scope,svm.name,ip.address"
    }
  }
}
```

Security

Accounts

List the accounts

You can retrieve a list of the accounts. You might do this to assess your security environment or before creating a new account.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/security/accounts

Processing type

Synchronous

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/security/accounts" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example


```
{
  "records": [
    {
      "owner": {
        "uuid": "642573a8-9d14-11ee-9330-005056aed3de",
        "name": "vs0",
        "_links": {
          "self": {
            "href": "/api/svm/svms/642573a8-9d14-11ee-9330-
005056aed3de"
          }
        }
      },
      "name": "vsadmin",
      "_links": {
        "self": {
          "href": "/api/security/accounts/642573a8-9d14-11ee-9330-
005056aed3de/vsadmin"
        }
      }
    },
    {
      "owner": {
        "uuid": "fdb6fe29-9d13-11ee-9330-005056aed3de",
        "name": "sti214nscluster-1"
      },
      "name": "admin",
      "_links": {
        "self": {
          "href": "/api/security/accounts/fdb6fe29-9d13-11ee-9330-
005056aed3de/admin"
        }
      }
    },
    {
      "owner": {
        "uuid": "fdb6fe29-9d13-11ee-9330-005056aed3de",
        "name": "sti214nscluster-1"
      },
      "name": "autosupport",
      "_links": {
        "self": {
          "href": "/api/security/accounts/fdb6fe29-9d13-11ee-9330-
005056aed3de/autosupport"
```

```
        }
      }
    }
  ],
  "num_records": 3,
  "_links": {
    "self": {
      "href": "/api/security/accounts"
    }
  }
}
```

Certificates and keys

List the installed certificates

You can list the certificates installed in your ONTAP cluster. You might do this to see if a particular certificate is available or to get the ID of a specific certificate.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/security/certificates

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
max_records	Query	No	Specify the number of records you want returned.

Curl example: Return three certificates

```
curl --request GET \
--location "https://$FQDN_IP/api/security/certificates?max_records=3" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example


```
{
  "records": [
    {
      "uuid": "dad822c2-573c-11ee-a310-005056aecc29",
      "name": "vs0_17866DB5C933E2EA",
      "_links": {
        "self": {
          "href": "/api/security/certificates/dad822c2-573c-11ee-a310-
005056aecc29"
        }
      }
    },
    {
      "uuid": "7d8e5570-573c-11ee-a310-005056aecc29",
      "name": "BuypassClass3RootCA",
      "_links": {
        "self": {
          "href": "/api/security/certificates/7d8e5570-573c-11ee-a310-
005056aecc29"
        }
      }
    },
    {
      "uuid": "7dbb2191-573c-11ee-a310-005056aecc29",
      "name": "EntrustRootCertificationAuthority",
      "_links": {
        "self": {
          "href": "/api/security/certificates/7dbb2191-573c-11ee-a310-
005056aecc29"
        }
      }
    }
  ],
  "num_records": 3,
  "_links": {
    "self": {
      "href": "/api/security/certificates?max_records=3"
    },
    "next": {
      "href": "/api/security/certificates?start.svm_id=sti214nscluster-
1&start.uuid=7dbb2191-573c-11ee-a310-005056aecc29&max_records=3"
    }
  }
}
```

Install a certificate

You can install a signed X.509 certificate in your ONTAP cluster. You might do this as part of configuring an ONTAP feature or protocol that requires strong authentication.

Before you begin

You must have the certificate you want to install. You should also make sure any intermediate certificates are installed as needed.

> Before using the JSON input examples included below, make sure to update the `public_certificate` value with the certificate for your environment.

Step 1: Install the certificate

You can issue an API call to install the certificate.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/certificates

Curl example: Install a root CA certificate at the cluster level

```
curl --request POST \
--location "https://$FQDN_IP/api/security/certificates" \
--include \
--header "Content-Type: application/json" \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example


```
{
    "type": "server_ca",
    "public_certificate":
    "-----BEGIN CERTIFICATE-----
MIID0TCCArkCFGYdznvTVvaY1VZPNfy4yCCyPph6MA0GCSqGSIb3DQEBCwUAMIGk
MQswCQYDVQQGEwJVUzELMAkGA1UECAwCTkMxDDAKBgNVBAcMA1JUUDEWMBQGA1UE
CgwNT05UQVAgRXhhbXBsZTETMBEGA1UECwwKT05UQVAgOS4xNDEcMBoGA1UEAwwT
Ki5vbnRhcC1leGFtcGxlLmNvbTEvMC0GCSqGSIb3DQEJARYgZGF2aWQucGV0ZXJz
b25Ab250YXAtZXhhbXBsZS5jb20wHhcNMjMxMDA1MTUyOTE4WhcNMjQxMDA0MTUy
OTE4WjCBpDELMAkGA1UEBhMCVVMxCzAJBgNVBAgMAk5DMQwwCgYDVQQHDANSVFAx
FjAUBgNVBAoMDU9OVEFQIEV4YW1wbGUxEzARBgNVBAsMCk9OVEFQIDkuMTQxHDAa
BgNVBAMMEyoub250YXAtZXhhbXBsZS5jb20xLzAtBgkqhkiG9w0BCQEWIGRhdmlk
LnBldGVyc29uQG9udGFwLWV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOC
AQ8AMIIBCgKCAQEAxQgy8mhb1Jhkf0D/MBodpzgW0aSp2jGbWJ+Zv2G8BXkp1762
dPHRkv1hnx9JvwkK4Dba05GiCiD5t3gjH/jUQMSFb+VwDbVmubVFnXjkm/4Q7sea
tMtA/ZpQdZbQFZ5RKtdWz7dzzPYEl2x8Q1Jc8Kh7NxERNMtgupGWZZn7mfXKYr4O
N/+vgahIhDibS8YK5rf1w6bfmrik9E2D+PEab9DX/1DL5RX4tZ1H2OkyN2UxoBR6
Fq7l6n1Hi/5yR0Oi1xStN6sO7EPoGak+KSlK41q+EcIKRo0bP4mEQp8WMjJuiTkb
5MmeYoIpWEUgJK7S0M6Tp/3bTh2CST3AWxiNxQIDAQABMA0GCSqGSIb3DQEBCwUA
A4IBAQABfBqOuROmYxdfrj93OyIiRoDcoMzvo8cHGNUsuhnlBDnL2O3qhWEs97s0
mIy6zFMGnyNYa0t4i1cFsGDKP/JuljmYHjvv+2lHWnxHjTo7AOQCnXmQH5swoDbf
o1Vjqz8Oxz+PRJ+PA3dF5/8zqaAR6QreAN/iFR++6nUq1sbbM7w03tthBVMgo/h1
E9I2jVOZsqMFujm2CYfMs4XkZtrYmN6nZA8JcUpDjIWcAVbQYurMnna9r42oS3GB
WB/FE9n+P+FfJyHJ93KGcCXbH5RF2pi3wLlHilbvVuCjLRrhJ8U20I5mZoiXvABc
IpYuBcuKXLwAarhDEacXttVjC+Bq
-----END CERTIFICATE-----"
}
```


Step 2: Confirm the certificate has been installed

Perform the workflow List the installed certificates and confirm the certificate is available.

RBAC

Prepare to use RBAC

You can use the ONTAP RBAC capability in several different ways depending on your environment. A few common scenarios are presented as workflows in this section. In each case the focus is on a specific security and administrative goal.

Before creating any roles and assigning a role to an ONTAP user account, you should prepare by reviewing the major security requirements and options presented below. Also make sure to review the general workflow concepts at Prepare to use the workflows.

What ONTAP release are you using?

The ONTAP release determines what REST endpoints and RBAC features are available.

Identify the protected resources and scope

You need to identify the resources or commands to be protected and the scope (cluster or SVM).

What access should the user have?

After identifying the resources and scope, you need to determine the access level to be granted.

How will the users access ONTAP?

The user can access ONTAP through the REST API or CLI or both.

Is one of the built-in roles sufficient or is a custom role needed?

It is more convenient to use an existing built-in role but you can create a new custom role if needed.

What type of role is needed?

Based on the security requirements and the ONTAP access, you need to choose whether to create a REST or traditional role.

Create roles

Limit access to SVM volume operations

You can define a role to restrict storage volume administration within an SVM.

About this workflow

A traditional role is first created to initially allow access to all the major volume administration functions except cloning. The role is defined with the following characteristics:

- Able to perform all CRUD volume operations including get, create, modify, and delete
- Cannot create a volume clone

You can then optionally update the role as needed. In this workflow, the role is changed in the second step to allow the user to create a volume clone.

Step 1: Create the role

You can issue an API call to create the RBAC role.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/security/roles" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "role1",
  "owner": {
    "name": "cluster-1",
    "uuid": "852d96be-f17c-11ec-9d19-005056bbad91"
  },
  "privileges": [
      { "path": "volume create", "access": "all" },
      { "path": "volume delete", "access": "all" }
    ]
}
```

Step 2: Update the role

You can issue an API call to update the existing role.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	This is the UUID of the SVM that contains the role definition.
$ROLE_NAME	Path	Yes	This is the name of the role within the SVM to be updated.

Curl example

```
curl --request POST \
--location
"https://$FQDN_IP/api/security/roles/$SVM_ID/$ROLE_NAME/priveleges" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "path": "volume clone",
  "access": "all"
}
```

Enable adminstration of data protection

You can provide a user with limited data protection capabilities.

About this workflow

The traditional role created is defined with the following characteristics:

- Able to create and delete snapshots as well as update SnapMirror relationships
- Cannot create or modify higher level objects such as volumes or SVMs

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/security/roles" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "role1",
  "owner": {
    "name": "cluster-1",
    "uuid": "852d96be-f17c-11ec-9d19-005056bbad91"
  },
  "privileges": [
      {"path": "volume snapshot create", "access": "all"},
      {"path": "volume snapshot delete", "access": "all"},
      {"path": "volume show", "access": "readonly"},
      {"path": "vserver show", "access": "readonly"},
      {"path": "snapmirror show", "access": "readonly"},
      {"path": "snapmirror update", "access": "all"}
  ]
}
```

Allow generation of ONTAP reports

You can create a REST role to provide users with the ability to generate ONTAP reports.

About this workflow

The role created is defined with the following characteristics:

- Able to retrieve all storage object information related to capacity and performance (such as volume, qtree, LUN, aggregates, node, and SnapMirror relationships)
- Cannot create or modify higher level objects (such as volumes or SVMs)

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/security/roles" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "rest_role1",
  "owner": {
    "name": "cluster-1",
    "uuid": "852d96be-f17c-11ec-9d19-005056bbad91"
  },
  "privileges": [
      {"path": "/api/storage/volumes", "access": "readonly"},
      {"path": "/api/storage/qtrees", "access": "readonly"},
      {"path": "/api/storage/luns", "access": "readonly"},
      {"path": "/api/storage/aggregates", "access": "readonly"},
      {"path": "/api/cluster/nodes", "access": "readonly"},
      {"path": "/api/snapmirror/relationships", "access": "readonly"},
      {"path": "/api/svm/svms", "access": "readonly"}
  ]
}
```

Create a user with a role

You can use this workflow to create a user with an associated REST role.

About this workflow

This workflow includes the typical steps needed to create a custom REST role and associate it with a new user account. Both the user and role have an SVM scope and are associated with a specific data SVM. Some of the steps may be optional or need to change depending on your environment.

Step 1: List the data SVMs in the cluster

Perform the following REST API call to list the SVMs in the cluster. The UUID and name of each SVM are provided in the output.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/svm/svms

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/svm/svms?order_by=name" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

After you finish

Select the desired SVM from the list where you will create the new user and role.

Step 2: List the users defined to the SVM

Perform the following REST API call to list the users defined in the SVM you selected. You can identify the SVM through the owner parameter.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/security/accounts

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/security/accounts?owner.name=dmp" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

After you finish

Based on the users already defined in the SVM, choose a unique name for the new user.

Step 3: List the REST roles defined to the SVM

Perform the following REST API call to list the roles defined in the SVM you selected. You can identify the SVM through the owner parameter.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/security/roles

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/security/roles?owner.name=dmp" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

After you finish

Based on the roles already defined in the SVM, choose a unique name for the new role.

Step 4: Create a custom REST role

Perform the following REST API call to a create a custom REST role in the SVM. The role initially has only one privilege which establishes a default access of **none** so that all access is denied.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/security/roles" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "dprole1",
  "owner": {
    "name": "dmp",
    "uuid": "752d96be-f17c-11ec-9d19-005056bbad91"
  },
  "privileges": [
      {"path": "/api", "access": "none"},
  ]
}
```

After you finish

Optionally perform step 3 again to display the new role. You can also display the roles at the ONTAP CLI.

Step 5: Update the role by adding more privileges

Perform the following REST API call to modify the role by adding privileges as needed.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/roles/{owner.uuid}/{name}/privileges

Additional input parameters for curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl example in this step.

Parameter	Type	Required	Description
$SVM_ID	Path	Yes	The UUID of the SVM that contains the role definition.
$ROLE_NAME	Path	Yes	The name of the role within the SVM to be updated.

Curl example

```
curl --request POST \
--location
"https://$FQDN_IP/api/security/roles/$SVM_ID/$ROLE_NAME/privileges" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "path": "/api/storage/volumes",
  "access": "readonly"
}
```

After you finish

Optionally perform step 3 again to display the new role. You can also display the roles at the ONTAP CLI.

Step 6: Create a user

Perform the following REST API call to a create a user account. The role **dprole1** created above is associated with the new user.

You can create the user without a role. In this case, the user is assigned a default role (either `admin` or `vsadmin`) depending on whether the user is defined with cluster or SVM scope. You'll need to modify the user to assign a different role.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/security/accounts

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/security/accounts" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "owner": {"uuid":"daf84055-248f-11ed-a23d-005056ac4fe6"},
  "name": "david",
  "applications": [
      {"application":"ssh",
       "authentication_methods":["password"],
       "second_authentication_method":"none"}
  ],
  "role":"dprole1",
  "password":"netapp123"
}
```

After you finish

You can sign in to the SVM management interface using the credentials for the new user.

Storage

List the aggregates

You can retrieve a list of aggregates in the cluster. You might do this to assess utilization and performance.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/storage/disks

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
node.name	Query	No	Can be used to identify the node each aggregate is attached to.

Curl example: return all aggregates with the default configuration values

```
curl --request GET \
--location "https://$FQDN_IP/api/storage/aggregates" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

Curl example: return all aggregates with a specific configuration value

```
curl --request GET \
--location "https://$FQDN_IP/api/storage/aggregates?fields=node.name" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example


```
{
  "records": [
    {
      "uuid": "760d8137-fc59-47da-906a-cc28db0a1c1b",
      "name": "sti214_vsim_sr027o_aggr1",
      "node": {
        "name": "sti214-vsim-sr027o"
      },
      "_links": {
        "self": {
          "href": "/api/storage/aggregates/760d8137-fc59-47da-906a-
cc28db0a1c1b"
        }
      }
    }
  ],
  "num_records": 1,
  "_links": {
    "self": {
      "href": "/api/storage/aggregates?fields=node.name"
    }
  }
}
```


List the disks

You can retrieve a list of disks in the cluster. You might do this to locate one or more spares to use as part of creating an aggregate.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/storage/disks

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
state	Query	No	Can be used to identify the spare disks available for new aggregates.

Curl example: return all the disks

```
curl --request GET \
--location "https://$FQDN_IP/api/storage/disks" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

Curl example: return spare disks

```
curl --request GET \
--location "https://$FQDN_IP/api/storage/disks?state=spare" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example


```
{
  "records": [
    {
      "name": "NET-1.20",
      "state": "spare",
      "_links": {
        "self": {
          "href": "/api/storage/disks/NET-1.20"
        }
      }
    },
    {
      "name": "NET-1.12",
      "state": "spare",
      "_links": {
        "self": {
          "href": "/api/storage/disks/NET-1.12"
        }
      }
    },
    {
      "name": "NET-1.7",
      "state": "spare",
      "_links": {
        "self": {
          "href": "/api/storage/disks/NET-1.7"
        }
      }
    }
  ],
  "num_records": 3,
  "_links": {
    "self": {
      "href": "/api/storage/disks?state=spare"
    }
  }
}
```


Support

EMS

Prepare to manage EMS support services

You can configure Event Management System (EMS) processing for an ONTAP cluster as well as retrieve EMS messages as needed.

Overview

There are several example workflows available that illustrate how to use the ONTAP EMS services. Before using the workflows and issuing any of the REST API calls, make sure to review Prepare to use the workflows.

If you use Python, also see the scripy events.py for examples of how to automate some of the EMS-related activities.

ONTAP REST API versus ONTAP CLI commands

For many tasks, using the ONTAP REST API requires fewer calls than the equivalent ONTAP CLI commands. The table below includes a list of API calls and the equivalent the CLI commands needed for each task.

ONTAP REST API	ONTAP CLI
GET /support/ems	`event config show`
POST /support/ems/destinations	1. `event notification destination create` 2. `event notification create`
`GET /support/ems/events`	`event log show`
`POST /support/ems/filters`	1. `event filter create -filter-name <filtername>` 2. `event filter rule add -filter-name <filtername>`

Related information

- Python script illustrating EMS
- ONTAP REST APIs: Automate Notification of High-Severity Events

List the EMS log events

You can retrieve all event notification messages or only those with specific characteristics.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/support/ems/events

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
fields	Query	No	Used to request specific fields to be included in the response.
max_records	Query	No	Can be used to limit the number of records returned in a single request.
log_message	Query	No	Used to search for a specific text value and only return the matching messages.
message.severity	Query	No	Limit the returned messages to those with a specific severity such as `alert`.

Curl example: Return the latest message and the name value

```
curl --request GET \
--location
"https://$FQDN_IP/api/support/ems/events?fields=message.name&max_records=1
" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

Curl example: Return a message containing specific text and severity

```
curl --request GET \
--location
"https://$FQDN_IP/api/support/ems/events?log_message=*disk*&message.severi
ty=alert" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example


```
{
  "records": [
    {
      "node": {
        "name": "malha-vsim1",
        "uuid": "da4f9e62-9de3-11ec-976a-005056b369de",
        "_links": {
          "self": {
            "href": "/api/cluster/nodes/da4f9e62-9de3-11ec-976a-
005056b369de"
          }
        }
      },
      "index": 4602,
      "time": "2022-03-18T06:37:46-04:00",
      "message": {
        "severity": "alert",
        "name": "raid.autoPart.disabled"
      },
      "log_message": "raid.autoPart.disabled: Disk auto-partitioning is
disabled on this system: the system needs a minimum of 4 usable internal
hard disks.",
      "_links": {
        "self": {
          "href": "/api/support/ems/events/malha-vsim1/4602"
        }
      }
    }
  ],
  "num_records": 1,
  "_links": {
    "self": {
      "href":
"/api/support/ems/events?log_message=*disk*&message.severity=alert&max_rec
ords=1"
    },
    "next": {
      "href": "/api/support/ems/events?start.keytime=2022-03-
18T06%3A37%3A46-04%3A00&start.node.name=malha-
vsim1&start.index=4602&log_message=*disk*&message.severity=alert"
    }
  }
}
```

Get the EMS configuration

You can retrieve the current EMS configuration for an ONTAP cluster. You might do this before updating the configuration or creating a new EMS notification.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/support/ems

Processing type

Synchronous

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/support/ems" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example

```
{
  "proxy_url": "https://proxyserver.mycompany.com",
  "proxy_user": "proxy_user",
  "mail_server": "mail@mycompany.com",
  "_links": {
    "self": {
      "href": "/api/resourcelink"
    }
  },
  "pubsub_enabled": "1",
  "mail_from": "administrator@mycompany.com"
}
```

Create an EMS notification

You can use the following workflow to create a new EMS notification destination to receive selected event messages.

Step 1: Configure the system-wide email settings

You can issue the following API call to configure the system-wide email settings.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
PATCH	/api/support/ems

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
mail_from	Query	Yes	Sets the `from` field in the notification email messages.
mail_server	Query	Yes	Configures the target SMTP mail server.

Curl example

```
curl --request PATCH \
--location
"https://$FQDN_IP/api/support/ems?mail_from=administrator@mycompany.com&ma
il_server=mail@mycompany.com" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

Step 2: Define a message filter

You can issue an API call to define a filter rule to match the messages.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/support/ems/filters

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
Filter	Body	Yes	Includes the values for the filter configuration.

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/support/ems/filters" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "test-filter",
  "rules.type": ["include"],
  "rules.message_criteria.severities": ["emergency"]
}
```

Step 3: Create a message destination

You can issue an API call to create a message destination.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/api/support/ems/destinations

Processing type

Synchronous

Additional input parameters for the Curl examples

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
Destination configuration	Body	Yes	Includes the values for the event destination.

Curl example

```
curl --request POST \
--location "https://$FQDN_IP/api/support/ems/destinations" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH" \
--data @JSONinput
```

JSON input example

```
{
  "name": "test-destination",
  "type": "email",
  "destination": "administrator@mycompany.com",
  "filters.name": ["important-events"]
}
```

SVM

List the SVMs

You can list the Storage Virtual Machines (SVMs) defined within an ONTAP cluster. You might do this as part of finding the identifier for a specific SVM or to assure name uniqueness before creating a new SVM.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/api/svm/svms

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/svm/svms" \
--include \
--header "Accept: */*" \
--header "Authorization: Basic $BASIC_AUTH"
```

JSON output example


```
{
  "records": [
    {
      "uuid": "71bd74f8-40dc-11ee-b51a-005056aee9fa",
      "name": "vs0",
      "_links": {
          "self": {
            "href": "/api/svm/svms/71bd74f8-40dc-11ee-b51a-005056aee9fa"
          }
      }
    }
  ],
  "num_records": 1,
  "_links": {
    "self": {
      "href": "/api/svm/svms"
    }
  }
}
```

Software tools

Python client library

Overview of the Python client library

The NetApp ONTAP Python client library is a package you can install and use to write scripts that access the ONTAP REST API. It provides support for several underlying services, including connection management, asynchronous processing, exception handling, and error messages. By using the Python client library, you can quickly develop robust code to support the automation of ONTAP deployments.

NetApp maintains a GitHub repository containing code samples and other helpful information. You can navigate to the *examples* folder to access samples using the Python client library.

Related information

- ONTAP REST Python GitHub repository
- ONTAP REST Python Client Library Examples

Prepare to use the Python client library

You should prepare the local runtime environment before using the Python client library.

Package name and version

The name of the Python client library package is **netapp-ontap**. The version associated with the package is a combination of the ONTAP major and minor version numbers the library was generated from, along with a minor version for the client within the ONTAP release. For example, valid version numbers include: 9.6.1, 9.6.2, and 9.7.1.

Installation

You must use pip to install the netapp_ontap package from the Python Package Index (PyPi) web site.

Packages and documentation by ONTAP release

Each ONTAP release beginning with 9.6 has a PyPI package and associated documentation. See Packages and documentation for more information. Installation requirements are include with each package and include different versions of the following:

- python
- requests
- requests-toolbelt
- marshmallow

Packages and documentation

The Python client library is available for each ONTAP release beginning with 9.6. You should access the PyPI package and documentation based on the ONTAP release you are using.

ONTAP 9.15.1

- PyPI: NetApp ONTAP 9.15.1
- NetApp PCL documentation for 9.15.1

ONTAP 9.14.1

- PyPI: NetApp ONTAP 9.14.1
- NetApp PCL documentation for 9.14.1

ONTAP 9.13.1

- PyPI: NetApp ONTAP 9.13.1
- NetApp PCL documentation for 9.13.1

ONTAP 9.12.1

- PyPI: NetApp ONTAP 9.12.1
- NetApp PCL documentation for 9.12.1

ONTAP 9.11.1

- PyPI: NetApp ONTAP 9.11.1
- NetApp PCL documentation for 9.11.1

ONTAP 9.10.1

- PyPI: NetApp ONTAP 9.10.1
- NetApp PCL documentation for 9.10.1

ONTAP 9.9.1

- PyPI: NetApp ONTAP 9.9.1
- NetApp PCL documentation for 9.9.1

ONTAP 9.8

- PyPI: NetApp ONTAP 9.8
- NetApp PCL documentation for 9.8

ONTAP 9.7

- PyPI: NetApp ONTAP 9.7
- NetApp PCL documentation for 9.7

ONTAP 9.6

- PyPI: NetApp ONTAP 9.6
- NetApp PCL documentation for 9.6

Script to retrieve the cluster configuration

The following script provides a simple example of how to use the Python client library. You can run the script using Python 3 at the CLI to retrieve the ONTAP cluster configuration.

```python
##-----------------------------------------------------------------------
#
# Description: Python script to retrieve the cluster configuration.
#
# Usage example:
#
# python3 get_cluster.py
#
#
# (C) Copyright 2024 NetApp, Inc.
#
# This sample code is provided AS IS, with no support or warranties of
# any kind, including but not limited for warranties of merchantability
# or fitness of any kind, expressed or implied. Permission to use,
# reproduce, modify and create derivatives of the sample code is granted
# solely for the purpose of researching, designing, developing and
# testing a software application product for use with NetApp products,
# provided that the above copyright notice appears in all copies and
# that the software application product is distributed pursuant to terms
# no less restrictive than those set forth herein.
#
##-----------------------------------------------------------------------
# For reading the password from the commandline
from getpass import getpass
# Global configuration for the library
from netapp_ontap import config
# Support for the connection to ONTAP
from netapp_ontap import HostConnection
# Specific API needed for this script
from netapp_ontap.resources import Cluster
# Create connection to the ONTAP management LIF
# (add verify=False if the certificate your cluster is serving is not trusted)
conn = HostConnection(
    "<mgmt_ip>", username="admin", password=getpass("ONTAP admin password: "),
)
# Set connection as the default for all API calls
config.CONNECTION = conn
# Create new cluster object
clus = Cluster()
# Issue REST API call
clus.get()
# Display the cluster configuration
```

```
43 print(clus)
```

Blog articles

There are several blog articles available to help you better understand how to use the Python client library.

Simplify ONTAP REST API Consumption with the Python client library

This blog provides a good introduction to the features of the ONTAP Python client library.

www.netapp.com/blog/simplify-ontap-rest-api-consumption

Getting Started with ONTAP REST API Python Client Library

This is a three-part series of blogs covering more details about the Python client library.

Part 1: netapp.io/2020/06/09/ontap-rest-api-python-client-library-pt1

Part 2: netapp.io/2020/06/09/ontap-rest-api-python-client-library-pt2/

Part 3: netapp.io/2020/06/09/ontap-rest-api-python-client-library-pt3

PowerShell toolkit

Overview of the PowerShell Toolkit

NetApp provides support for using PowerShell to administer your ONTAP storage systems.

PowerShell

PowerShell is a program from Microsoft that you can use for task automation and configuration management. It includes a command line shell environment as well as a scripting language.

NetApp ONTAP PowerShell Toolkit

The NetApp.ONTAP PowerShell toolkit includes the PowerShell module for NetApp ONTAP. The toolkit supports ONTAP running in a variety of environments, including NetApp AFF and FAS systems, commodity hardware, and the cloud. The module includes over 2,400 cmdlets which collectively support storage administration on Windows hosts.

Download and install the ONTAP PowerShell Toolkit

There are two options available to download and install the NetApp ONTAP PowerShell toolkit.

NetApp support

You can download the PowerShell toolkit from the NetApp support site:

NetApp.ONTAP PowerShell Toolkit

PowerShell Gallery

You can download the PowerShell toolkit from PowerShell Gallery:

NetApp Manageability SDK

The NetApp Manageability SDK provides a set of ONTAPI API calls for developing applications to monitor and manage your ONTAP storage. Together with the OnCommand Workflow Automation package, the SDK supports your efforts to automate the manage of your ONTAP systems.

While the NetApp Manageability SDK and OnCommand Workflow Automation continued to be supported, the ONTAP REST API is the preferred and strategic technology to use when automating your ONTAP systems. See ONTAPI disablement for more information.

Download the SDK

You can download the NetApp Manageability SDK from the NetApp Support Site. The SDK supports several languages on the client side, including: Python, PowerShell, C, C++, Java, C#, VB.Net, and Ruby. Make sure to review the Interoperability Matrix Tool for information about the NetApp Manageability SDK and how it is supported with your version of ONTAP.

Use OnCommand Workflow Automation

You can also use the API provided with the SDK to automate management tasks without writing any scripts. OnCommand Workflow Automation (OnCommand WFA) provides several prepackaged workflows to deploy and run the management tasks. You can download the OnCommand WFA package from the NetApp Storage Automation Store.

Related information

- NetApp Support Site
- NetApp Interoperability Matrix Tool
- NetApp Manageability SDK documentation
- OnCommand Workflow Automation documentation resources
- NetApp Automation Store

Migrate from ONTAPI to the REST API

ONTAPI disablement

The ONTAPI API (ZAPI) is the original set of proprietary calls included with the NetApp ONTAP software. The API is provided through the Network Manageability SDK and supports the automation of data storage administration and management tasks. The ONTAPI interface will be disabled in future versions of ONTAP. If you are using ONTAPI, you should plan your migration to the ONTAP REST API.

Related information

- Understand the ONTAP automation options
- CPC-00410 Deferral of ONTAPI (ZAPI) End-of-Availability Announcement
- FAQs on ZAPI to ONTAP REST API transformation for CPC

Migration considerations

Before migrating to the ONTAP REST API from either the ONTAPI API (ZAPI) or ONTAP CLI, there are several issues you should consider.

General design differences

The ONTAP REST API and command line interface have a fundamentally different designs. The CLI commands and parameters do not map directly to the REST API calls. And even where there might be a similarity, the details of the input parameters can be different. For example, numeric units might be specified in bytes or using a suffix (such as KB). See Input variables controlling an API request and API reference for more information.

Data SVMs exposed through the REST API

ONTAP supports several types of storage virtual machines (SVMs). However, only the data SVMs are directly exposed through the ONTAP REST API. The configuration information describing the cluster and nodes is available through the REST API, however the cluster and nodes are not treated as separate SVMs.

Access the ONTAP CLI through the REST API

To assist ONTAPI API and CLI users in their transition to the ONTAP REST API, ONTAP provides a REST endpoint to access the ONTAP CLI. You can use this passthrough feature to execute any CLI command. Use of the REST endpoint is returned in AutoSupport data so NetApp can identify gaps in the REST API and make improvements in future ONTAP releases.

To issue a CLI command, you must make a REST API call that is properly formed based on rules regarding the following:

- Resource paths
- Field names
- HTTP methods

The base resource path for CLI access is `/private/cli`. Refer to the ONTAP API online documentation page for details about accessing the CLI through the REST API. NetApp also maintains a GitHub repository containing code samples and other helpful information. See ONTAP REST Python GitHub repository - CLI passthrough samples for more information.

Changes to SnapDiff availability in ONTAPI

Beginning with ONTAP 9.10.1, the SnapDiff v1 and v2 ONTAPI calls cannot be invoked. Any third-party application that invokes SnapDiff v1 or v2 ONTAPI calls will not function beginning with ONTAP 9.10.1. ONTAP users should verify that their backup application supports the SnapDiff v3 REST calls before upgrading to ONTAP 9.10.1.

SnapDiff API availability across ONTAP releases is defined as follows:

- ONTAP 9.7 and earlier releases: v1 and v2 (ONTAPI only)
- ONTAP 9.8 – 9.9.1: v1, v2 and v3 (both ONTAPI and REST API)
- ONTAP 9.10.1: v3 only (REST API only)

Also see the ONTAP Release Notes for more information.

Submit your ONTAPI to REST API gaps

NetApp is committed to supporting our customers with their migration from ONTAPI to the ONTAP REST API. If you notice something missing in the REST API, please let us know. You can submit these gaps and any other ideas at the ONTAPI to REST API page.

ONTAPI to REST API mapping

The ONTAP REST API includes functionality that is equivalent to ONTAPI in most areas. NetApp provides documentation that describes the mapping from the ONTAPI calls to the equivalent REST API calls.

The API mapping documentation is dependent on the ONTAP release:

- ONTAP 9.15.1
- ONTAP 9.14.1
- ONTAP 9.13.1
- ONTAP 9.12.1
- ONTAP 9.11.1
- ONTAP 9.10.1
- ONTAP 9.9.1
- ONTAP 9.8

Performance counters

The ONTAP Counter Manager maintains extensive information about the performance of each ONTAP system. It exports this data as a set of *performance counters* you can use to

assess the performance of your ONTAP system and help meet your performance goals.

Access the ONTAP performance counters

You can access the ONTAP performance counters using two different APIs as well as through the ONTAP command line interface.

> The ONTAP REST API is the preferred and strategic option when automating the administration of your ONTAP deployments.

ONTAPI API

The ONTAPI API is available with the NetApp Network Manageability SDK. When using ONTAPI, the performance counters are defined within a collection of objects. Each object corresponds to a physical or virtual component of the system. There can be one or more instances of each object based on the system configuration.

For example, if your ONTAP system has four physical disks, there will be four instances of the `disk` object, each with its own set of performance counters. You can use ONTAPI to access the individual counters for each disk instance.

ONTAP REST API

Beginning with ONTAP 9.11.1, you can also access the performance data through the REST API. In this case, the performance counters are organized in tables which are equivalent to the ONTAPI objects. Each table row is equivalent to an instance of an ONTAPI object.

For example, if your ONTAP system has four physical disks, the `disk` table will contain four rows. Each of the rows can be accessed individually and includes its own set of performance counters available as fields or columns in the row.

Prepare to use the REST API

You should prepare before using the ONTAP REST API to access the performance counters.

Performance counters organized in tables

A subset of the ONTAPI objects is available through the ONTAP REST API and presented as tables. For example, the ONTAPI **hostadapter** object is presented through the REST API as the **host_adpater** table. Each host adapter in the system is a row with its own set of performance counters.

Instance name	Performance counters					
host_adapter_1	total_read_ops_1	total_write_ops_1	bytes_read_1	bytes_written_1	max_link_data_rate_1	rscn_count_1
host_adapter_2	total_read_ops_2	total_write_ops_2	bytes_read_2	bytes_written_2	max_link_data_rate_2	rscn_count_2
host_adapter_3	total_read_ops_3	total_write_ops_3	bytes_read_3	bytes_written_3	max_link_data_rate_3	rscn_count_3

Summary of the REST endpoints

There are four major endpoints available to access the ONTAP performance counters and related tables.

> Each of the REST endpoints provides read-only access and only supports the **GET** HTTP method. See the API reference for more information.

- **/cluster/counter/tables**

  Returns a collection of counter tables and their schema definitions.

- **/cluster/counter/tables/{name}**

  Returns information about a single named counter table.

- **/cluster/counter/tables/{counter_name}/rows**

  Returns a collection of rows from a named counter table.

- **/cluster/counter/tables/{counter_name}/rows/{id}**

  Returns a specific row from a named counter table.

Migrating from ONTAPI to the REST API

NetApp provides extensive support for migrating your automation code from ONTAPI to the ONTAP REST API. This includes mapping documentation to identity the equivalent performance counter table available in the REST API for a given ONTAPI object.

See the appropriate mapping documentation based on the ONTAP release you are using:

- ONTAP 9.15.1 performance counter mapping
- ONTAP 9.14.1 performance counter mapping
- ONTAP 9.13.1 performance counter mapping
- ONTAP 9.12.1 performance counter mapping
- ONTAP 9.11.1 performance counter mapping

Get started with the ONTAP REST API

The following examples illustrate how to use REST API to access the ONTAP performance counters. This includes retrieving a list of the available tables and exploring the table structure.

Before you begin

Review the following information before using the examples.

ONTAP credentials

You'll need an ONTAP administrator account including the password.

Cluster management IP

You'll need the cluster management IP address configured for your ONTAP system.

All API calls use the GET method

All of the examples included below can only be used to retrieve information with the HTTP GET method.

Variable substitution

Each curl example includes one or more variables as indicated with capitals and bracketed text. Make sure to replace these variables with actual values as appropriate for your environment.

Examples match endpoints

The sequence of examples below illustrates how to use the REST endpoints available for retrieving the performance counters. See Summary of the REST endpoints for more information.

Example 1: All performance counter tables

You can use this REST API call to discover all the available counter manager tables.

Curl example

```
curl --request GET --user admin:<PASSWORD>
'https://<ONTAP_IP_ADDRESS>/api/cluster/counter/tables'
```

JSON output example


```
{
  "records": [
    {
      "name": "copy_manager",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/copy_manager"
        }
      }
    },
    {
      "name": "copy_manager:constituent",
      "_links": {
        "self": {
          "href":
"/api/cluster/counter/tables/copy_manager%3Aconstituent"
        }
      }
    },
    {
      "name": "disk",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/disk"
        }
      }
    },
    {
      "name": "disk:constituent",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/disk%3Aconstituent"
        }
      }
    },
    {
      "name": "disk:raid_group",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/disk%3Araid_group"
        }
      }
    },
    {
```

```
      "name": "external_cache",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/external_cache"
        }
      }
    },
    {
      "name": "fcp",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/fcp"
        }
      }
    },
    {
      "name": "fcp:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/fcp%3Anode"
        }
      }
    },
    {
      "name": "fcp_lif",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/fcp_lif"
        }
      }
    },
    {
      "name": "fcp_lif:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/fcp_lif%3Anode"
        }
      }
    },
    {
      "name": "fcp_lif:port",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/fcp_lif%3Aport"
        }
      }
```

```
    },
    {
      "name": "fcp_lif:svm",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/fcp_lif%3Asvm"
        }
      }
    },
    {
      "name": "fcvi",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/fcvi"
        }
      }
    },
    {
      "name": "headroom_aggregate",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/headroom_aggregate"
        }
      }
    },
    {
      "name": "headroom_cpu",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/headroom_cpu"
        }
      }
    },
    {
      "name": "host_adapter",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/host_adapter"
        }
      }
    },
    {
      "name": "iscsi_lif",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/iscsi_lif"
```

```
        }
      }
    },
    {
      "name": "iscsi_lif:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/iscsi_lif%3Anode"
        }
      }
    },
    {
      "name": "iscsi_lif:svm",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/iscsi_lif%3Asvm"
        }
      }
    },
    {
      "name": "lif",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/lif"
        }
      }
    },
    {
      "name": "lif:svm",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/lif%3Asvm"
        }
      }
    },
    {
      "name": "lun",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/lun"
        }
      }
    },
    {
      "name": "lun:constituent",
      "_links": {
```

```
      "self": {
        "href": "/api/cluster/counter/tables/lun%3Aconstituent"
      }
    }
  },
  {
    "name": "lun:node",
    "_links": {
      "self": {
        "href": "/api/cluster/counter/tables/lun%3Anode"
      }
    }
  },
  {
    "name": "namespace",
    "_links": {
      "self": {
        "href": "/api/cluster/counter/tables/namespace"
      }
    }
  },
  {
    "name": "namespace:constituent",
    "_links": {
      "self": {
        "href": "/api/cluster/counter/tables/namespace%3Aconstituent"
      }
    }
  },
  {
    "name": "nfs_v4_diag",
    "_links": {
      "self": {
        "href": "/api/cluster/counter/tables/nfs_v4_diag"
      }
    }
  },
  {
    "name": "nic_common",
    "_links": {
      "self": {
        "href": "/api/cluster/counter/tables/nic_common"
      }
    }
  },
  {
```

```
      "name": "nvmf_lif",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/nvmf_lif"
        }
      }
    },
    {
      "name": "nvmf_lif:constituent",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/nvmf_lif%3Aconstituent"
        }
      }
    },
    {
      "name": "nvmf_lif:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/nvmf_lif%3Anode"
        }
      }
    },
    {
      "name": "nvmf_lif:port",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/nvmf_lif%3Aport"
        }
      }
    },
    {
      "name": "object_store_client_op",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/object_store_client_op"
        }
      }
    },
    {
      "name": "path",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/path"
        }
      }
```

```
    },
    {
      "name": "processor",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/processor"
        }
      }
    },
    {
      "name": "processor:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/processor%3Anode"
        }
      }
    },
    {
      "name": "qos",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/qos"
        }
      }
    },
    {
      "name": "qos:constituent",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/qos%3Aconstituent"
        }
      }
    },
    {
      "name": "qos:policy_group",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/qos%3Apolicy_group"
        }
      }
    },
    {
      "name": "qos_detail",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/qos_detail"
```

```
        }
      }
    },
    {
      "name": "qos_detail_volume",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/qos_detail_volume"
        }
      }
    },
    {
      "name": "qos_volume",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/qos_volume"
        }
      }
    },
    {
      "name": "qos_volume:constituent",
      "_links": {
        "self": {
          "href":
"/api/cluster/counter/tables/qos_volume%3Aconstituent"
        }
      }
    },
    {
      "name": "qtree",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/qtree"
        }
      }
    },
    {
      "name": "qtree:constituent",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/qtree%3Aconstituent"
        }
      }
    },
    {
      "name": "svm_cifs",
```

```
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/svm_cifs"
        }
      }
    },
    {
      "name": "svm_cifs:constituent",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/svm_cifs%3Aconstituent"
        }
      }
    },
    {
      "name": "svm_cifs:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/svm_cifs%3Anode"
        }
      }
    },
    {
      "name": "svm_nfs_v3",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/svm_nfs_v3"
        }
      }
    },
    {
      "name": "svm_nfs_v3:constituent",
      "_links": {
        "self": {
          "href":
"/api/cluster/counter/tables/svm_nfs_v3%3Aconstituent"
        }
      }
    },
    {
      "name": "svm_nfs_v3:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/svm_nfs_v3%3Anode"
        }
      }
```

```
    },
    {
      "name": "svm_nfs_v4",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/svm_nfs_v4"
        }
      }
    },
    {
      "name": "svm_nfs_v41",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/svm_nfs_v41"
        }
      }
    },
    {
      "name": "svm_nfs_v41:constituent",
      "_links": {
        "self": {
          "href":
"/api/cluster/counter/tables/svm_nfs_v41%3Aconstituent"
        }
      }
    },
    {
      "name": "svm_nfs_v41:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/svm_nfs_v41%3Anode"
        }
      }
    },
    {
      "name": "svm_nfs_v42",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/svm_nfs_v42"
        }
      }
    },
    {
      "name": "svm_nfs_v42:constituent",
      "_links": {
        "self": {
```

```
          "href":
"/api/cluster/counter/tables/svm_nfs_v42%3Aconstituent"
        }
      }
    },
    {
      "name": "svm_nfs_v42:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/svm_nfs_v42%3Anode"
        }
      }
    },
    {
      "name": "svm_nfs_v4:constituent",
      "_links": {
        "self": {
          "href":
"/api/cluster/counter/tables/svm_nfs_v4%3Aconstituent"
        }
      }
    },
    {
      "name": "svm_nfs_v4:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/svm_nfs_v4%3Anode"
        }
      }
    },
    {
      "name": "system",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/system"
        }
      }
    },
    {
      "name": "system:constituent",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/system%3Aconstituent"
        }
      }
    },
```

```
    {
      "name": "system:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/system%3Anode"
        }
      }
    },
    {
      "name": "token_manager",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/token_manager"
        }
      }
    },
    {
      "name": "volume",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/volume"
        }
      }
    },
    {
      "name": "volume:node",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/volume%3Anode"
        }
      }
    },
    {
      "name": "volume:svm",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/volume%3Asvm"
        }
      }
    },
    {
      "name": "wafl",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/wafl"
        }
```

```
      }
    },
    {
      "name": "wafl_comp_aggr_vol_bin",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/wafl_comp_aggr_vol_bin"
        }
      }
    },
    {
      "name": "wafl_hya_per_aggregate",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/wafl_hya_per_aggregate"
        }
      }
    },
    {
      "name": "wafl_hya_sizer",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/wafl_hya_sizer"
        }
      }
    }
  ],
  "num_records": 71,
  "_links": {
    "self": {
      "href": "/api/cluster/counter/tables"
    }
  }
}
```


Example 2: High-level information about a specific table

You can use this REST API call to display the description and metadata for a specific table. The output includes the purpose of the table and what type of data each performance counter contains. The **host_adapter** table is used in this example.

Curl example

```
curl --request GET --user admin:<PASSWORD>
'https://<ONTAP_IP_ADDRESS>/api/cluster/counter/tables/host_adapter'
```

JSON output example


```
{
  "name": "host_adapter",
  "description": "The host_adapter table reports activity on the Fibre
Channel, Serial Attached SCSI, and parallel SCSI host adapters the
storage system uses to connect to disks and tape drives.",
  "counter_schemas": [
    {
      "name": "bytes_read",
      "description": "Bytes read through a host adapter",
      "type": "rate",
      "unit": "per_sec"
    },
    {
      "name": "bytes_written",
      "description": "Bytes written through a host adapter",
      "type": "rate",
      "unit": "per_sec"
    },
    {
      "name": "max_link_data_rate",
      "description": "Max link data rate in Kilobytes per second for a
host adapter",
      "type": "raw",
      "unit": "kb_per_sec"
    },
    {
      "name": "node.name",
      "description": "System node name",
      "type": "string",
      "unit": "none"
    },
    {
      "name": "rscn_count",
      "description": "Number of RSCN(s) received by the FC HBA",
      "type": "raw",
      "unit": "none"
    },
    {
      "name": "total_read_ops",
      "description": "Total number of reads on a host adapter",
      "type": "rate",
      "unit": "per_sec"
    },
    {
```

```
      "name": "total_write_ops",
      "description": "Total number of writes on a host adapter",
      "type": "rate",
      "unit": "per_sec"
    }
  ],
  "_links": {
    "self": {
      "href": "/api/cluster/counter/tables/host_adapter"
    }
  }
}
```


Example 3: All rows in a specific table

You can use this REST API call to view all the rows in a table. This indicates what instances of the Counter Manager objects exist.

Curl example

```
curl --request GET --user admin:<PASSWORD>
'https://<ONTAP_IP_ADDRESS>/api/cluster/counter/tables/host_adapter/row
s'
```

JSON output example


```
{
  "records": [
    {
      "id": "dmp-adapter-01",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/host_adapter/rows/dmp-
adapter-01"
        }
      }
    },
    {
      "id": "dmp-adapter-02",
      "_links": {
        "self": {
          "href": "/api/cluster/counter/tables/host_adapter/rows/dmp-
adapter-02"
        }
      }
    }
  ],
  "num_records": 2,
  "_links": {
    "self": {
      "href": "/api/cluster/counter/tables/host_adapter/rows"
    }
  }
}
```


Example 4: Single row in a specific table

You can use this REST API call to view performance counter values for a specific counter manager instance in the table. In this example, the performance data for one of the host adapters is requested.

Curl example

```
curl --request GET --user admin:<PASSWORD>
'https://<ONTAP_IP_ADDRESS>/api/cluster/counter/tables/host_adapter/row
s/dmp-adapter-01'
```

JSON output example

```
{
  "counter_table": {
    "name": "host_adapter"
  },
  "id": "dmp-adapter-01",
  "properties": [
    {
      "name": "node.name",
      "value": "dmp-node-01"
    }
  ],
  "counters": [
    {
      "name": "total_read_ops",
      "value": 25098
    },
    {
      "name": "total_write_ops",
      "value": 48925
    },
    {
      "name": "bytes_read",
      "value": 1003799680
    },
    {
      "name": "bytes_written",
      "value": 6900961600
    },
    {
      "name": "max_link_data_rate",
      "value": 0
    },
    {
      "name": "rscn_count",
      "value": 0
    }
  ],
  "_links": {
    "self": {
      "href": "/api/cluster/counter/tables/host_adapter/rows/dmp-
adapter-01"
    }
  }
}
```

Tools and software

NetApp provides sample Python scripts and other related software to support your migration from ONTAPI to the ONTAP REST API. The most important of these samples are described below.

All the Python code samples are available at the NetApp ONTAP REST Python GitHub repository.

ONTAPI usage reporting tool

The ONTAPI usage reporting tool is designed to help NetApp professional services, customers, and partners identify the ONTAPI usage in their ONTAP environment. Scripts are provided for three different use cases as described in the table below.

Script	Description
apache_scraper.py	An Apache log scraper to find the ONTAPI calls issued against the ONTAP nodes
session_stats.py	A CLI script to retrieve session statistics data from ONTAP
zapi_to_rest.py	A script to extract the REST details of the ONTAPI calls and attributes passed

You can access the ONTAPI usage reporting tool to get started. Also see a Demo about the reporting tool and how to use it.

Private CLI passthrough

The REST API provides broad coverage of the features and facilities available with ONTAP. However, there may be instances when direct access to the ONTAP CLI through the REST API can be useful.

For an introduction to this feature, see Access the ONTAP CLI through the REST API. For the Python samples see REST CLI passthrough samples.

Python client library

The Python client library is a package you can install and use to access the ONTAP REST API with Python. It allows you to quickly develop robust code for the automation of your ONTAP deployments.

For an introduction to the Python client library, see Overview of the Python client library. For the Python samples see Python client library examples.

ONTAP PowerShell Toolkit

The NetApp.ONTAP PowerShell Toolkit enhances your local PowerShell environment with a module that includes over 2,400 cmdlets. It allows you to quickly develop code for your Windows host to automate the ONTAP deployments. For more information, see Overview of the PowerShell Toolkit.

Blog articles

There are several blog articles available to help you better understand how to migrate from ONTAPI to the ONTAP REST API.

ONTAPI to REST Mapping

NetApp provides support for moving from the proprietary ONTAPI API to the ONTAP REST API through mapping documentation.

netapp.io/2020/12/17/ontapi-to-rest-mapping

Transform your Automation to ONTAP REST API from ONTAPI

There are several technologies available to help you transform your ONTAP automation environment based on the REST API.

www.netapp.com/blog/transform-automation-ontap-rest-api

Using the private CLI passthrough with the ONTAP REST API

To help CLI and ONTAP users transition to the ONTAP REST API, ONTAP provides a private REST API endpoint that can be used to access any CLI command.

https://netapp.io/2020/11/09/private-cli-passthrough-ontap-rest-api

Transitioning from ONTAPI using ONTAPI Usage Reporting Tool

NetApp provides a tool to help customers and partners transition to the ONTAP REST API.

netapp.io/2022/03/21/transitioning-from-ontapizapi-using-ontapi-usage-reporting-tool

API reference

The API reference contains details about the ONTAP REST API calls, including the HTTP methods, input parameters, and responses. This complete reference is helpful when developing automation applications using the REST API.

You can access the REST API reference documentation at one of several sites based on the ONTAP release. A copy of the equivalent documentation is also available through the Swagger UI at your local ONTAP system.

Access the ONTAP API reference documentation online

You can access a copy of the ONTAP REST API reference documentation through the links provided below. The documentation is maintained by ONTAP release.

- ONTAP 9.15.1
- ONTAP 9.14.1
- ONTAP 9.13.1
- ONTAP 9.12.1
- ONTAP 9.11.1
- ONTAP 9.10.1
- ONTAP 9.9.1
- ONTAP 9.8
- ONTAP 9.7
- ONTAP 9.6

Access the ONTAP API reference documentation through the Swagger UI

You can access the ONTAP REST API documentation through the Swagger UI at your local ONTAP system.

Before you begin

You must have the following:

- IP address or host name of the ONTAP cluster management LIF
- User name and password for an account with authority to access the ONTAP REST API

Steps

1. Type the URL in your browser and press **Enter**:

   `https://<ip_address>/docs/api`

2. Sign in using the ONTAP account.

   The ONTAP API documentation page is displayed with the API calls organized in major resource categories at the bottom.

3. As an example of an individual API call, scroll down to the **cluster** category and click **GET /cluster**.

Learn more

There are several additional resources available to help you automate the deployment and administration of your ONTAP storage systems.

Blog articles

- A good summary of the current ONTAP automation technologies.

  New normal for automation

- An introduction to accessing and using the Python samples scripts at GitHub for the ONTAP REST API.

  Get started with sample scripts at GitHub

- The CLI passthrough provides a technique for executing ONTAP CLI commands using the REST API.

  Using the private CLI passthrough with the ONTAP REST API

- If you're new to using Ansible for ONTAP automation, this is a good place to start.

  Getting Started with NetApp and Ansible: Install Ansible

- You can explore using the ONTAP REST API to manage file security and permissions.

  Simplified management of file-security permissions with ONTAP REST APIs

- You can monitor ONTAP events to stay aware of the system activity. The configuration and management of these events can be automated using the REST API.

  ONTAP REST APIs: Automate Notification of High-Severity Events

- You can use the REST API to configure roles and access levels as part of an RBAC security environment.

  Role-based Access Control (RBAC) using ONTAP REST APIs

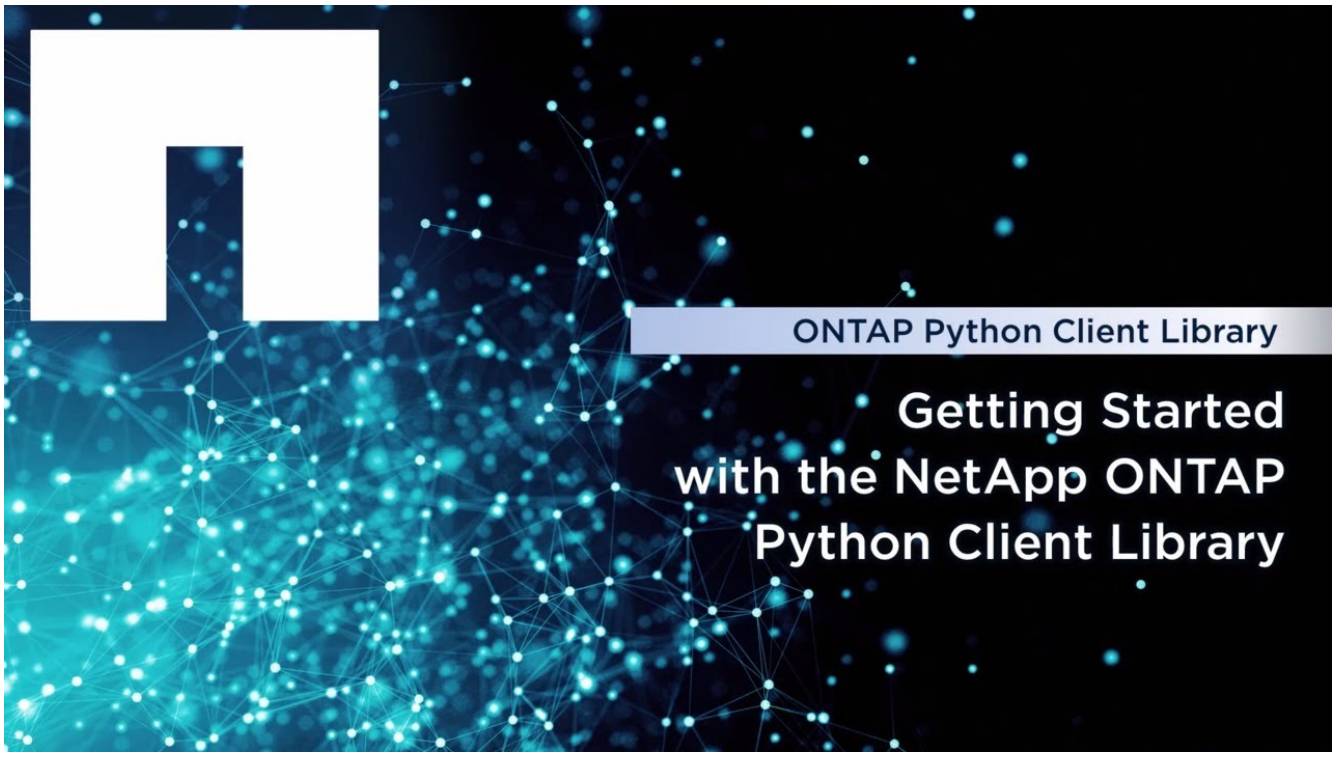
- NetApp provides Python sample scripts to use the ONTAP REST API.

  ONTAP REST API Python Sample Scripts Now Available on GitHub!

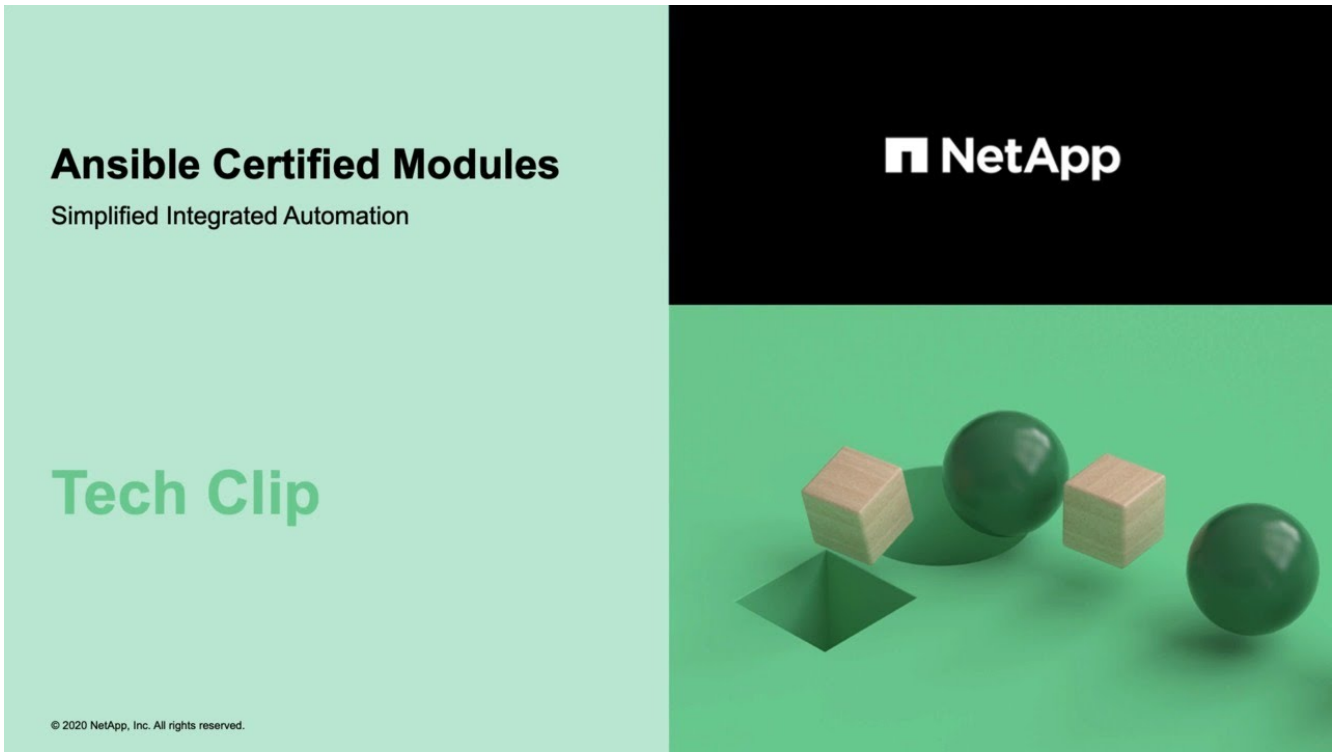
- Coffee breaks with REST (6 episodes).
  - Basics of ONTAP REST APIs
  - Features of ONTAP REST APIs
  - Getting Hands-on with ONTAP REST using Postman
  - ONTAPI (ZAPI) Reporting tool
  - Private CLI Passthrough
  - 5 magical features that make ONTAP storage automation simple!

Videos

- A good introduction to the NetApp Python client library and how to get started writing code using the library.



- A look at the Ansible Certified Modules.



- A collection of videos at NetApp TechComm TV.

  Automate NetApp ONTAP Management

Technical training and events

- Insight 2022 presentation (26 minutes).

  Modernize your ONTAP Storage Management with ONTAP REST API

- Insight 2021 presentation (31 minutes).

  NetApp ONTAP: Save time and simplify using REST APIs

- NetApp Learning Services.

  Automate Storage Administration Using ONTAP REST API and Ansible

NetApp Knowledge Base

- If you encounter an issue with the ONTAP REST API, you can report it to NetApp.

  How to report issues on ONTAP REST API and ONTAP REST API Python client library

- If you identify a functional gap in the ONTAP REST API, you can request a new feature for the API.

  How to request a feature for ONTAP REST API

Legal notices

Legal notices provide access to copyright statements, trademarks, patents, and more.

Copyright

https://www.netapp.com/company/legal/copyright/

Trademarks

NETAPP, the NETAPP logo, and the marks listed on the NetApp Trademarks page are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.

https://www.netapp.com/company/legal/trademarks/

Patents

A current list of NetApp owned patents can be found at:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Privacy policy

https://www.netapp.com/company/legal/privacy-policy/